다이나믹 프로그래밍 2

최백준 choi@startlink.io

이동하기

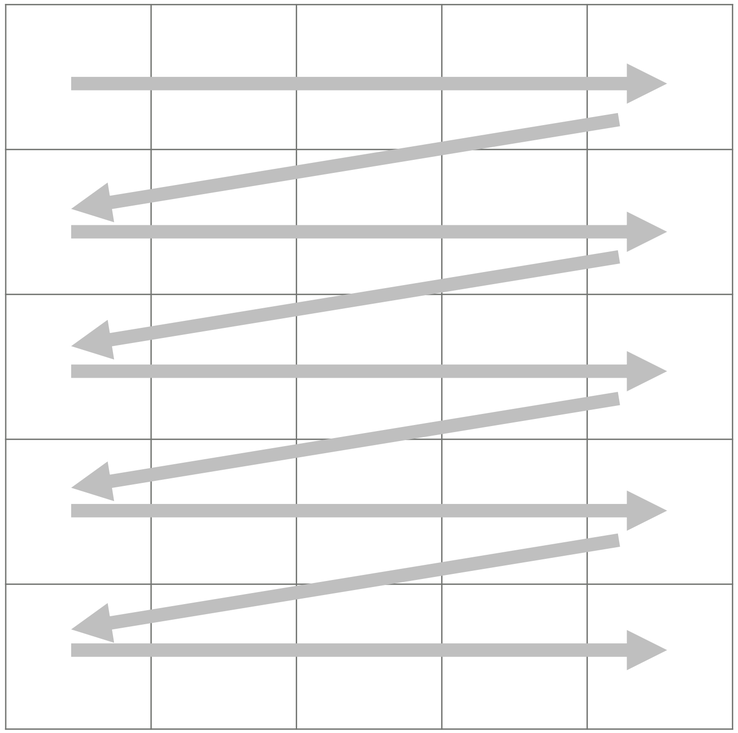
https://www.acmicpc.net/problem/11048

- 준규는 N×M 크기의 미로에 갇혀있다
- 미로는 1×1크기의 방으로 나누어져 있고, 각 방에는 사탕이 놓여져 있다
- 미로의 가장 왼쪽 윗 방은 (1, 1)이고, 가장 오른쪽 아랫 방은 (N, M)이다
- 준규는 현재 (1, 1)에 있고, (N, M)으로 이동하려고 한다
- 준규가 (i, j)에 있으면, (i+1, j), (i, j+1), (i+1, j+1)로 이동할 수 있고, 각 방을 방문할 때마다 방에 놓여져있는 사탕을 모두 가져갈 수 있다
- 또, 미로 밖으로 나갈 수는 없다
- 준규가 (N, M)으로 이동할 때, 가져올 수 있는 사탕 개수의 최대값

이동하기

https://www.acmicpc.net/problem/11048

- 항상 아래와 오른쪽으로만 갈 수 있다.
- (i,j)에서 가능한 이동: (i+1, j), (i, j+1), (i+1, j+1)

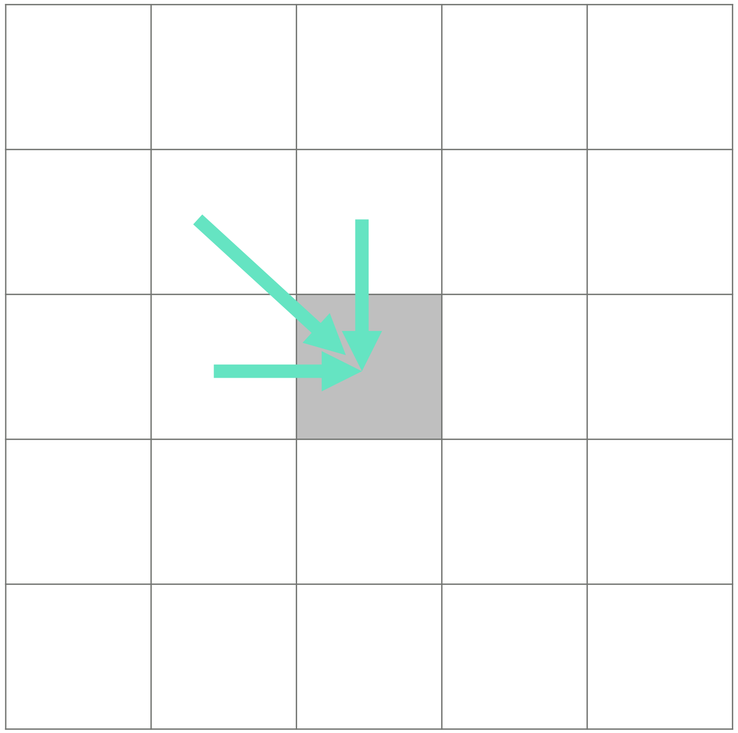


방법 1

이동하기

https://www.acmicpc.net/problem/11048

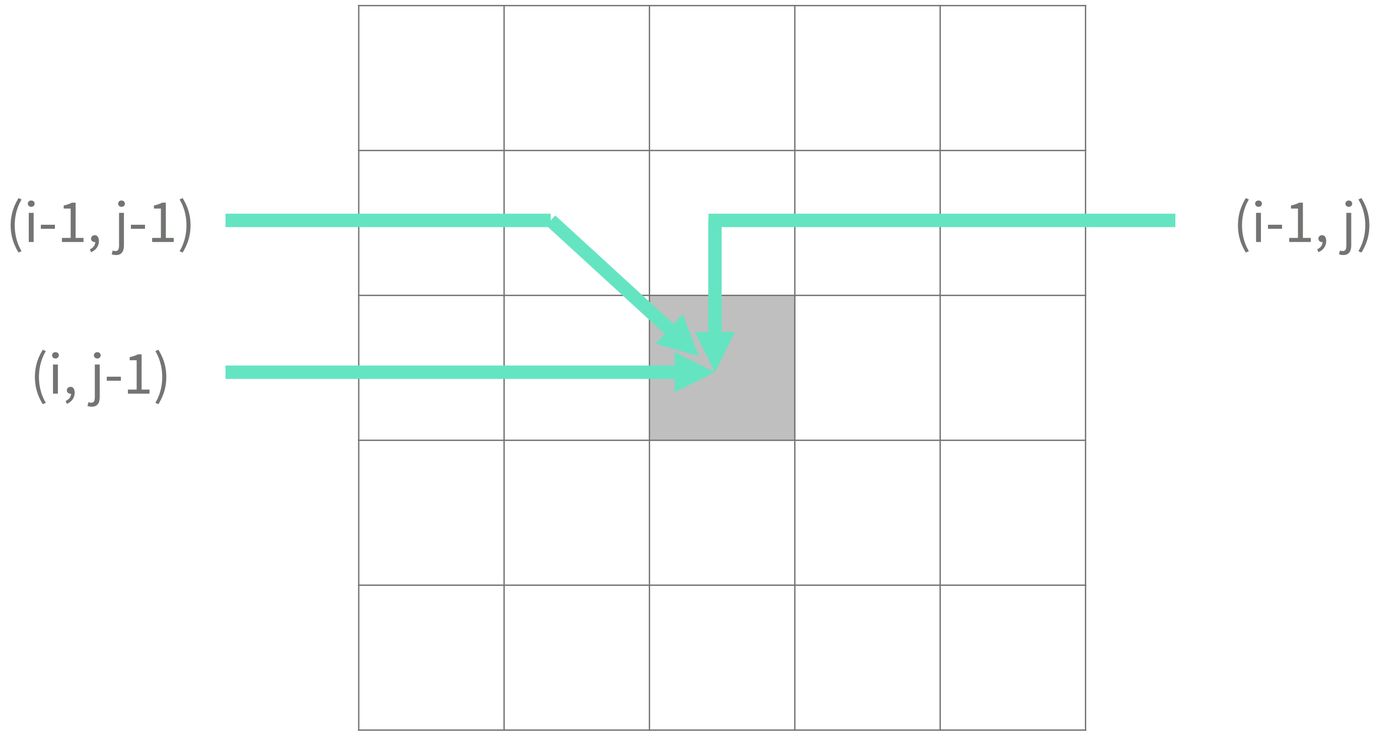
- 항상 아래와 오른쪽으로만 갈 수 있다.
- (i,j)에서 가능한 이동: (i+1, j), (i, j+1), (i+1, j+1)



이동하기

https://www.acmicpc.net/problem/11048

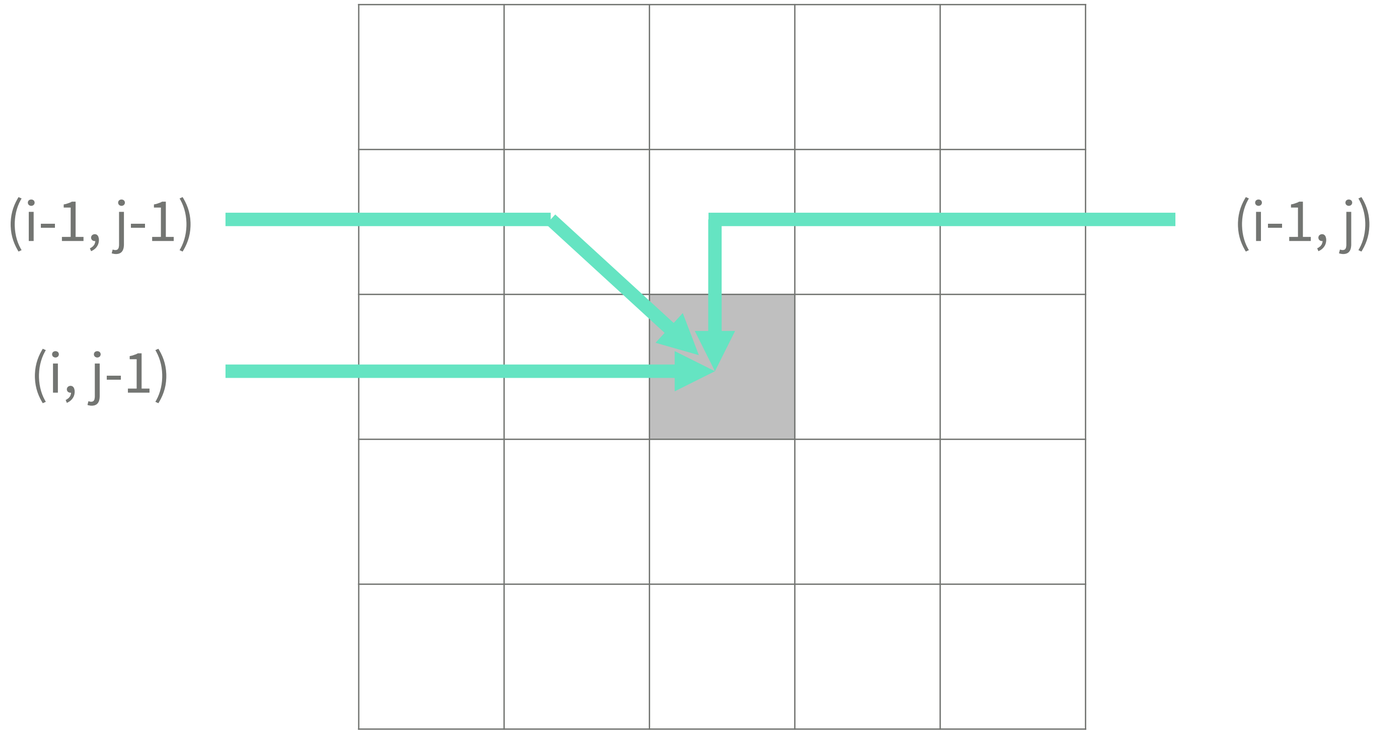
- 항상 아래와 오른쪽으로만 갈 수 있다.
- (i,j)에서 가능한 이동: (i+1, j), (i, j+1), (i+1, j+1)



이동하기

https://www.acmicpc.net/problem/11048

- D[i][j] = (i, j)로 이동할 때 가져올 수 있는 최대 사탕 개수
- D[i][j] = Max(D[i-1][j-1], D[i][j-1], D[i-1][j]) + A[i][j]



이동하기

https://www.acmicpc.net/problem/11048

```
for (int i=1; i<=n; i++) {
    for (int j=1; j<=m; j++) {
        d[i][j] = max3(d[i-1][j],d[i][j-1],d[i-1][j-1])+a[i][j];
    }
}
```

이동하기

https://www.acmicpc.net/problem/11048

```
for (int i=1; i<=n; i++) {
    for (int j=1; j<=m; j++) {
        d[i][j] = max3(d[i-1][j],d[i][j-1],d[i-1][j-1])+a[i][j];
    }
}
```

- i-1, j-1 범위 검사를 하지 않은 이유
- i = 1, j =1인 경우
- i = 1인 경우
- j = 1인 경우

이동하기

https://www.acmicpc.net/problem/11048

```
for (int i=1; i<=n; i++) {
    for (int j=1; j<=m; j++) {
        d[i][j] = max3(d[i-1][j],d[i][j-1],d[i-1][j-1])+a[i][j];
    }
}
```

- i-1, j-1 범위 검사를 하지 않은 이유
- i = 1, j =1인 경우: d[i-1][j], d[i][j-1], d[i-1][j-1]은 0이기 때문
- i = 1인 경우: d[i-1][j] = 0 < d[i][j-1] 이기 때문
- j = 1인 경우: d[i][j-1] = 0 < d[i-1][j] 이기 때문

이동하기

https://www.acmicpc.net/problem/11048

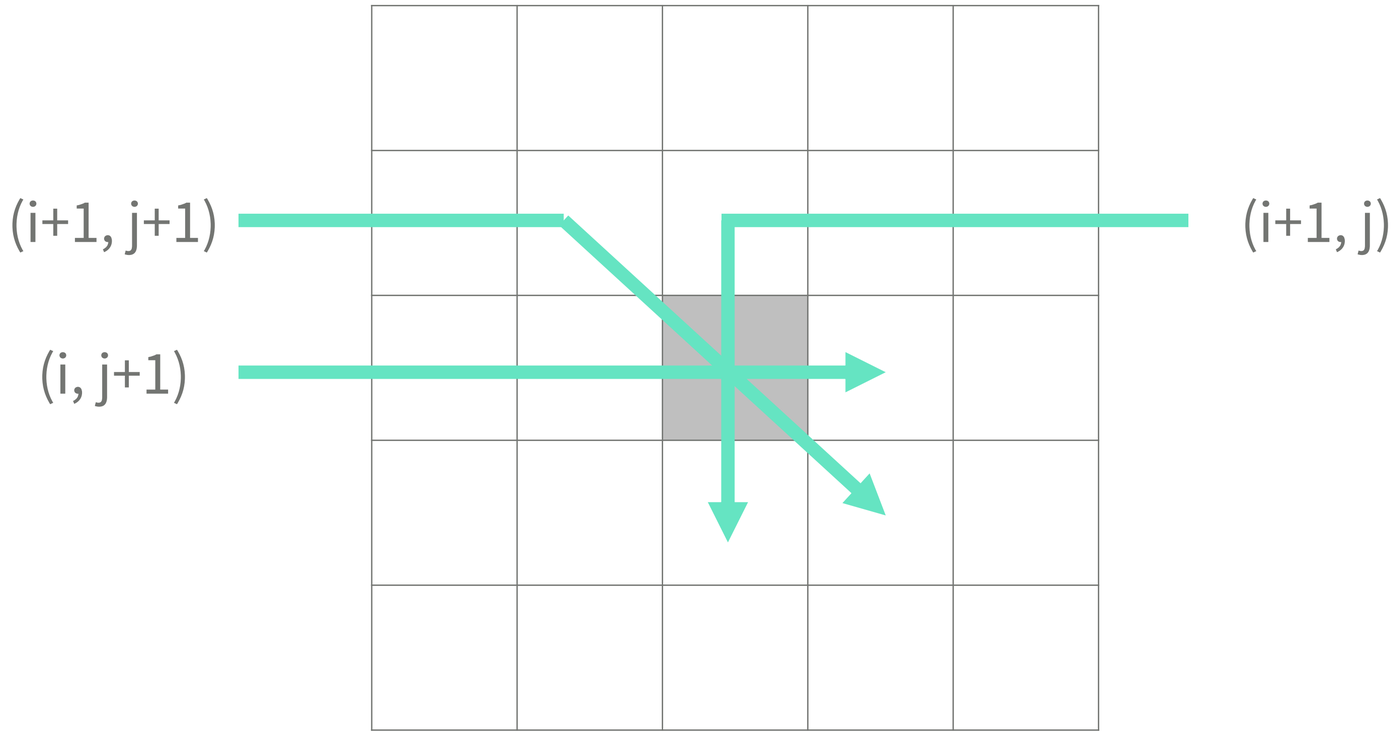
- 소스: http://codeplus.codes/0c449422d9f74402a8cc01bb57d65f57

방법 2

이동하기

https://www.acmicpc.net/problem/11048

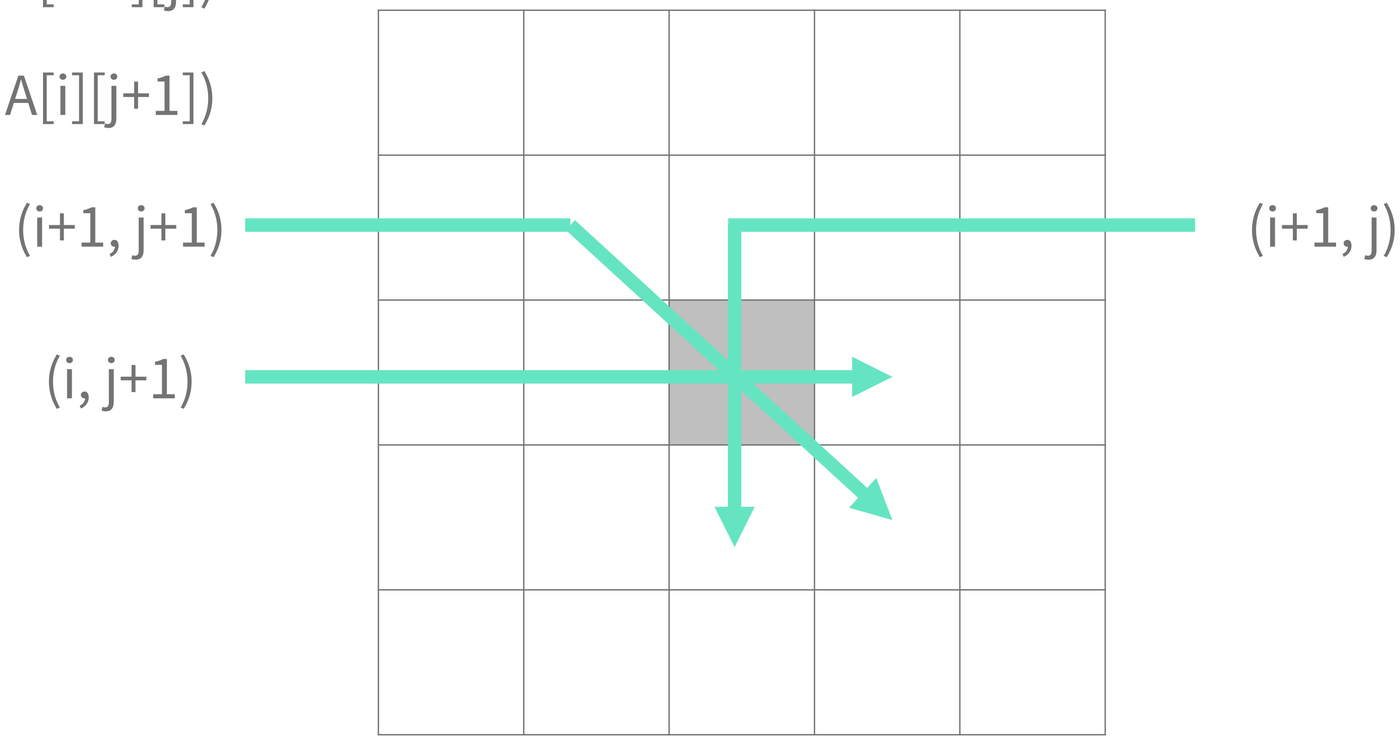
- 항상 아래와 오른쪽으로만 갈 수 있다.
- (i,j)에서 가능한 이동: (i+1, j), (i, j+1), (i+1, j+1)



이동하기

https://www.acmicpc.net/problem/11048

- D[i][j] = (i, j)로 이동할 때 가져올 수 있는 최대 사탕 개수
- D[i+1][j+1] = max(D[i+1][j+1], D[i][j] + A[i+1][j+1])
- D[i+1][j] = max(D[i+1][j], D[i][j] + A[i+1][j])
- D[i][j+1] = max(D[i][j+1], D[i][j] + A[i][j+1])



이동하기

https://www.acmicpc.net/problem/11048

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

이동하기

https://www.acmicpc.net/problem/11048

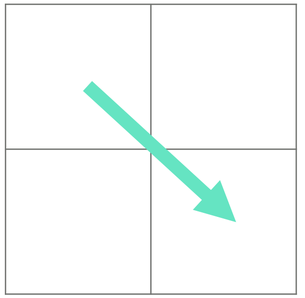
- 소스: http://codeplus.codes/47f630a02a80426da2ad43eacae9e1c1

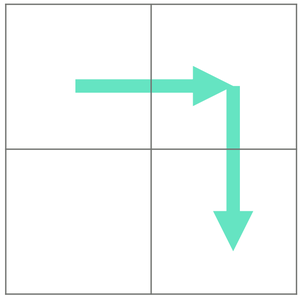
방법 3

이동하기

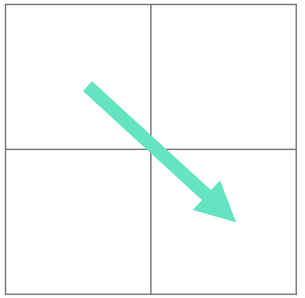
https://www.acmicpc.net/problem/11048

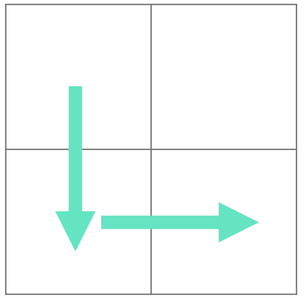
- 대각선 이동은 처리하지 않아도 된다
- 대각선 이동은 다른 2가지를 포함한 방법보다 항상 작거나 같다
- A[i][j] + A[i+1][j+1] ≤ A[i][j] + A[i][j+1] + A[i+1][j+1]





- A[i][j] + A[i+1][j+1] ≤ A[i][j] + A[i+1][j] + A[i+1][j+1]





이동하기

https://www.acmicpc.net/problem/11048

```
for (int i=1; i<=n; i++) {
    for (int j=1; j<=m; j++) {
        d[i][j] = max(d[i-1][j],d[i][j-1])+a[i][j];
    }
}
```

이동하기

https://www.acmicpc.net/problem/11048

- 소스: http://codeplus.codes/e21e683837504d19a147bf027a7f5503

방법 4

이동하기

https://www.acmicpc.net/problem/11048

- 재귀 함수를 이용해서도 구현할 수 있다
- D[i][j] = (i, j)로 이동할 때 가져올 수 있는 최대 사탕 개수
- D[i][j] = max(D[i][j-1], D[i-1][j]) + A[i][j]
- 식이 달라지는 것이 아니고 구현 방식이 달라지는 것이다

이동하기

https://www.acmicpc.net/problem/11048

```
// Bottom-up
for (int i=1; i<=n; i++) {
    for (int j=1; j<=m; j++) {
        d[i][j] = max(d[i-1][j],d[i][j-1]) + a[i][j];
    }
}
cout << d[n][m] << '\n';
```

이동하기

https://www.acmicpc.net/problem/11048

```
// Bottom-up
for (int i=1; i<=n; i++) {
    for (int j=1; j<=m; j++) {
        d[i][j] = max(d[i-1][j],d[i][j-1]) + a[i][j];
    }
}
cout << d[n][m] << '\n';
// Top-down
int go(int i, int j) {
}
```

이동하기

https://www.acmicpc.net/problem/11048

```
// Bottom-up
for (int i=1; i<=n; i++) {
    for (int j=1; j<=m; j++) {
        d[i][j] = max(d[i-1][j],d[i][j-1]) + a[i][j];
    }
}
cout << d[n][m] << '\n';
// Top-down
int go(int i, int j) {
    d[i][j] = max(d[i-1][j],d[i][j-1]) + a[i][j];
}
```

이동하기

https://www.acmicpc.net/problem/11048

```
// Bottom-up
for (int i=1; i<=n; i++) {
    for (int j=1; j<=m; j++) {
        d[i][j] = max(d[i-1][j],d[i][j-1]) + a[i][j];
    }
}
cout << d[n][m] << '\n';
// Top-down
int go(int i, int j) {
    d[i][j] = max(go(i-1,j), go(i,j-1)) + a[i][j];
}
```

이동하기

https://www.acmicpc.net/problem/11048

```
// Bottom-up
for (int i=1; i<=n; i++) {
    for (int j=1; j<=m; j++) {
        d[i][j] = max(d[i-1][j],d[i][j-1]) + a[i][j];
    }
}
cout << d[n][m] << '\n';
// Top-down
int go(int i, int j) {
    d[i][j] = max(go(i-1,j), go(i,j-1)) + a[i][j];
    return d[i][j];
}
```

이동하기

https://www.acmicpc.net/problem/11048

```
// Bottom-up
for (int i=1; i<=n; i++) {
    for (int j=1; j<=m; j++) {
        d[i][j] = max(d[i-1][j],d[i][j-1]) + a[i][j];
    }
}
cout << d[n][m] << '\n';
// Top-down
int go(int i, int j) {
    d[i][j] = max(go(i-1,j), go(i,j-1)) + a[i][j];
    return d[i][j];
}
cout << d[n][m] << '\n';
```

이동하기

https://www.acmicpc.net/problem/11048

```
// Bottom-up
for (int i=1; i<=n; i++) {
    for (int j=1; j<=m; j++) {
        d[i][j] = max(d[i-1][j],d[i][j-1]) + a[i][j];
    }
}
cout << d[n][m] << '\n';
// Top-down
int go(int i, int j) {
    d[i][j] = max(go(i-1,j), go(i,j-1)) + a[i][j];
    return d[i][j];
}
cout << go(n,m) << '\n';
```

이동하기

https://www.acmicpc.net/problem/11048

```
// Bottom-up
for (int i=1; i<=n; i++) {
    for (int j=1; j<=m; j++) {
    }
}
cout << d[n][m] << '\n';
// Top-down
int go(int i, int j) {
    if (i < 1 || j < 1) return 0;
    d[i][j] = max(go(i-1,j), go(i,j-1)) + a[i][j];
    return d[i][j];
}
cout << go(n,m) << '\n';
```

이동하기

https://www.acmicpc.net/problem/11048

```
int go(int i, int j) {
    if (i < 1 || j < 1) {
        return 0;
    }
    if (d[i][j] > 0) {
        return d[i][j];
    }
    d[i][j] = max(go(i-1, j), go(i, j-1)) + a[i][j];
    return d[i][j];
}
```

이동하기

https://www.acmicpc.net/problem/11048

- 시간 초과를 받게 된다.
- 사탕의 개수가 0보다 크거나 같기 때문에
- d[i][j]에 들어있는 값이 0인 경우가 답을 구했는데, 0일 수도 있다.
- 따라서, 같은 문제의 정답을 여러 번 구하게 된다.

이동하기

https://www.acmicpc.net/problem/11048

- 답이 음수인 경우는 절대로 없기 때문에, 배열을 -1로 초기화 하고 사용한다.

이동하기

https://www.acmicpc.net/problem/11048

```
int go(int i, int j) {
    if (i < 1 || j < 1) {
        return 0;
    }
    if (d[i][j] >= 0) {
        return d[i][j];
    }
    d[i][j] = max(go(i-1, j), go(i, j-1)) + a[i][j];
    return d[i][j];
}
```

이동하기

https://www.acmicpc.net/problem/11048

- 소스: http://codeplus.codes/6ec22bbcce094f2ca8e0aea256a1dcde

방법 5

이동하기

https://www.acmicpc.net/problem/11048

- 방법 1~4의 점화식은 모두 같았는데
- 구현 방식, 식을 채우는 순서만 조금씩 달랐다

이동하기

https://www.acmicpc.net/problem/11048

- 점화식을 조금 바꿔서 세워보자
- D[i][j] = (i, j)에서 이동을 시작했을 때, 가져올 수 있는 최대 사탕 개수
- 지금까지의 점화식
- D[i][j] = (i, j)로 이동했을 때, 가져올 수 있는 최대 사탕 개수

이동하기

https://www.acmicpc.net/problem/11048

- 점화식을 조금 바꿔서 세워보자
- D[i][j] = (i, j)에서 이동을 시작했을 때, 가져올 수 있는 최대 사탕 개수
- 도착(N, M)으로 정해져 있는데, 시작(i, j)을 이동시키는 방식
- 지금까지의 점화식
- D[i][j] = (i, j)로 이동했을 때, 가져올 수 있는 최대 사탕 개수
- 시작은 (1, 1)로 정해져 있고, 도착 (i, j)을 이동시 키는 방식

이동하기

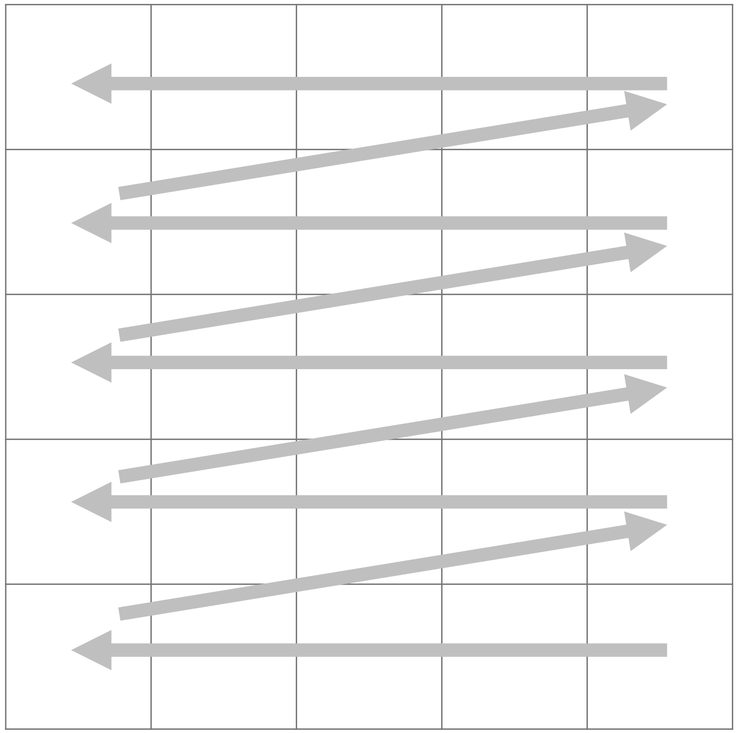
https://www.acmicpc.net/problem/11048

- D[i][j] = (i, j)에서 이동을 시작했을 때, 가져올 수 있는 최대 사탕 개수
- D[i][j] = max(D[i+1][j], D[i][j+1]) + A[i][j]

이동하기

https://www.acmicpc.net/problem/11048

- 항상 아래와 오른쪽으로만 갈 수 있다.
- (i,j)에서 가능한 이동: (i+1, j), (i, j+1), (i+1, j+1)



이동하기

https://www.acmicpc.net/problem/11048

```
int go(int x, int y) {
    if (x > n || y > m) return 0;
    if (d[x][y] >= 0) return d[x][y];
    d[x][y] = max(go(x+1,y), go(x,y+1)) + a[x][y];
    return d[x][y];
}
```

이동하기

https://www.acmicpc.net/problem/11048

- D[i][j] = (i, j)에서 이동을 시작했을 때, 가져올 수 있는 최대 사탕 개수
- D[i][j] = max(D[i+1][j], D[i][j+1]) + A[i][j]
- 정답은 D[1][1]에 있다.
- 즉, go(1, 1)을 호출해서 답을 구해야 한다.

이동하기

https://www.acmicpc.net/problem/11048

- 소스: http://codeplus.codes/5a7141a6fcd14f4eb6a1e8deeb512d19

문제

점프 점프

https://www.acmicpc.net/problem/11060

- 크기가 1×N인 미로가 있고, 미로의 각 칸에는 A_i가 있다. $N \leq 1{,}000$
- i번째 칸에서 1, 2, …, A_i개의 칸만큼 점프할 수 있다.
- $A_3 = 3$인 경우 3번 칸에서 점프할 수 있는 곳은 4, 5, 6번 칸이다.
- N번 칸에 최소 몇 번의 점프를 해야 갈 수 있는지 구하는 문제

점프 점프

https://www.acmicpc.net/problem/11060

- D[i] = i번 칸에 도착할 수 있는 최소 점프 횟수

점프 점프

https://www.acmicpc.net/problem/11060

- i번 칸에 오기 전에 어디서 왔을지 알 수 없다. 모든 j번 칸에 대해서 조사해봐야 한다.
- j번 → i번으로 갈 수 있으려면 i-j ≤ A[j]가 되어야 한다.

점프 점프

https://www.acmicpc.net/problem/11060

- i번 칸에 오기 전에 어디서 왔을지 알 수 없다. 모든 j번 칸에 대해서 조사해봐야 한다.
- j번 → i번으로 갈 수 있으려면 $i-j \leq A[j]$가 되어야 한다.
- $D[i] = \min(D[j]) + 1$ ($j < i$, $i-j \leq A[j]$)

점프 점프

https://www.acmicpc.net/problem/11060

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

점프 점프

https://www.acmicpc.net/problem/11060

- 소스: http://codeplus.codes/93f70cd978c140dcb31589ebc492fe8e

점프 점프

https://www.acmicpc.net/problem/11060

- i번 칸에 최소 점프로 온 다음, 어디로 갈지 알 수 없다. 점프할 수 있는 칸의 수인 모든 j에 대해서 조사해 봐야 한다.
- i번 → i+j번으로 갈 수 있으려면 $j \leq A[i]$가 되어야 한다.

점프 점프

https://www.acmicpc.net/problem/11060

- i번 칸에 최소 점프로 온 다음, 어디로 갈지 알 수 없다. 점프할 수 있는 칸의 수인 모든 j에 대해서 조사해 봐야 한다.
- i번 → i+j번으로 갈 수 있으려면 $j \leq A[i]$가 되어야 한다.
- $D[i+j] = \min(D[i]) + 1$ $(1 \leq j \leq A[i])$

점프 점프

https://www.acmicpc.net/problem/11060

```
d[0] = 0;
for (int i=0; i<n-1; i++) {
    if (d[i] == -1) continue;
    for (int j=1; j<=a[i]; j++) {
        if (i+j >= n) {
            break;
        }
        if (d[i+j] == -1 || d[i+j] > d[i] + 1) {
            d[i+j] = d[i] + 1;
        }
    }
}
```

점프 점프

https://www.acmicpc.net/problem/11060

- 소스: http://codeplus.codes/c1f83d33a8b84119aa5eec5d4811f1b3

퇴사 2

https://www.acmicpc.net/problem/15486

- N+1일이 되는 날 퇴사를 하려고 한다 ($1 \leq N \leq 1,500,000$)
- 남은 N일 동안 최대한 많은 상담을 하려고 한다
- 하루에 하나의 상담을 할 수 있고
- i일에 상담을 하면, T[i]일이 걸리고 P[i]원을 번다

퇴사 2

https://www.acmicpc.net/problem/15486

- D[i] = i일이 되었을 때, 최대 수익

퇴사 2

https://www.acmicpc.net/problem/15486

- D[i] = i일이 되었을 때, 최대 수익
- i일에 있는 상담을 하는 경우
- i일에 있는 상담을 하지 않는 경우

퇴사 2

https://www.acmicpc.net/problem/15486

- D[i] = i일이 되었을 때, 최대 수익
- i일에 있는 상담을 하는 경우
  - i+T[i]일이 된다.
  - 수익은 P[i]이 늘어난다.
- i일에 있는 상담을 하지 않는 경우
  - i+1일이 된다.
  - 수익은 그대로이다.

퇴사 2

https://www.acmicpc.net/problem/15486

- D[i] = i일이 되었을 때, 최대 수익
- i일에 있는 상담을 하는 경우: D[i+T[i]] = max(D[i+T[i]], D[i]+P[i])
  - i+T[i]일이 된다.
  - 수익은 P[i]이 늘어난다.
- i일에 있는 상담을 하지 않는 경우: D[i+1] = max(D[i+1], D[i])
  - i+1일이 된다.
  - 수익은 그대로이다.

퇴사 2

https://www.acmicpc.net/problem/15486

- 정답은 D[N+1]에 들어있다.

퇴사 2

https://www.acmicpc.net/problem/15486

- 소스: http://codeplus.codes/8455d593f4ba442c8a130495c08dc85a

팰린드롬?

https://www.acmicpc.net/problem/10942

- 어떤 수열의 부분 수열이 팰린드롬인지 확인하는 문제
- 팰린드롬인지 확인하는데 걸리는 시간 : O(N)
- 질문이 M개면 O(MN)이라는 시간이 걸림
- $1 \leq M \leq 1{,}000{,}000$, $1 \leq N \leq 2{,}000$

팰린드롬?

https://www.acmicpc.net/problem/10942

- D[i][j] = A[i] ~ A[j]가 팰린드롬이면 1, 아니면 0
- 길이가 1인 부분 수열은 반드시 팰린드롬이다
  - D[i][i] = 1
- 길이가 2인 부분 수열은 두 수가 같을 때만 팰린드롬이다
  - D[i][i+1] = 1 (A[i] == A[i+1])
  - D[i][i+1] = 0 (A[i] != A[i+1])

팰린드롬?

https://www.acmicpc.net/problem/10942

- D[i][j] = A[i] ~ A[j]가 팰린드롬이면 1, 아니면 0
- 길이가 1인 부분 수열은 반드시 팰린드롬이다
  - D[i][i] = 1
- 길이가 2인 부분 수열은 두 수가 같을 때만 팰린드롬이다
  - D[i][i+1] = 1 (A[i] == A[i+1])
  - D[i][i+1] = 0 (A[i] != A[i+1])
- A[i] ~ A[j]가 팰린드롬이 되려면, A[i] == A[j] 이어야 하고, A[i+1] ~ A[j-1]이 팰린드롬이어야 한다
  - D[i][j] = 1 (A[i] == A[j] && D[i+1][j-1] == 1)

팰린드롬?

https://www.acmicpc.net/problem/10942

- 일반적인 방식으로 배열을 채우지 않기 때문에 재귀 호출을 사용하는 것이 좋다

팰린드롬?

https://www.acmicpc.net/problem/10942

```
int go(int i, int j) {
    if (i == j) {
        return 1;
    } else if (i+1 == j) {
        if (a[i] == a[j]) return 1;
        else return 0;
    }
    if (d[i][j] >= 0) return d[i][j];
    if (a[i] != a[j]) return d[i][j] = 0;
    else return d[i][j] = go(i+1,j-1);
}
```

팰린드롬?

https://www.acmicpc.net/problem/10942

- 재귀 호출을 사용하지 않고도 풀 수 있다
- 길이가 1인 D[i][j]를 채우고
- 2인 것을 채우고
- 3인 것을 채우고
- …
- N-1인 것을 채우는 방식을 이용하면
- for문으로도 채울 수 있다

팰린드롬?

https://www.acmicpc.net/problem/10942

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

팰린드롬?

https://www.acmicpc.net/problem/10942

- Bottom-up 소스: http://codeplus.codes/388df4598f4a467cb53779ef1bb31985
- Top-down 소스: http://codeplus.codes/16b8b3cecc694fee8f67ebdcd60adb02

1, 2, 3 더하기

https://www.acmicpc.net/problem/9095

- 정수 n을 1, 2, 3의 합으로 나타내는 방법의 수를 구하는 문제
- 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

https://www.acmicpc.net/problem/9095

- D[i] = i를 1, 2, 3의 조합으로 나타내는 방법의 수
- D[i] = D[i-1] + D[i-2] + D[i-3]

1, 2, 3 더하기

https://www.acmicpc.net/problem/9095

0

1

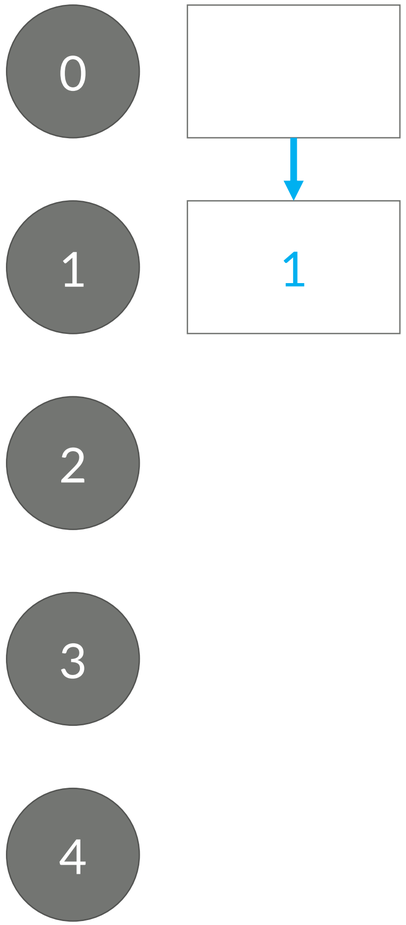
2

3

4

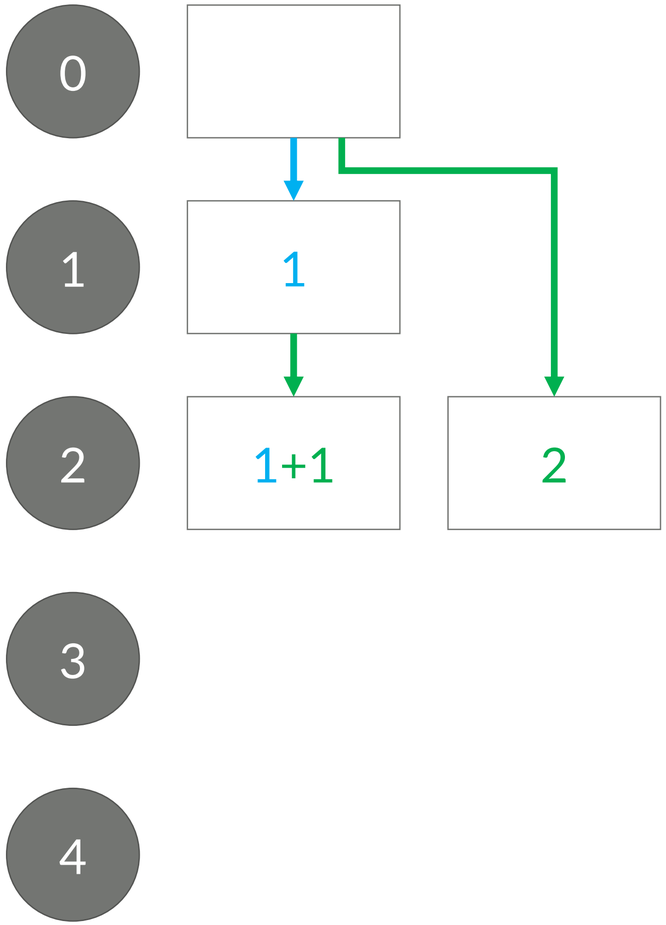
1, 2, 3 더하기

https://www.acmicpc.net/problem/9095



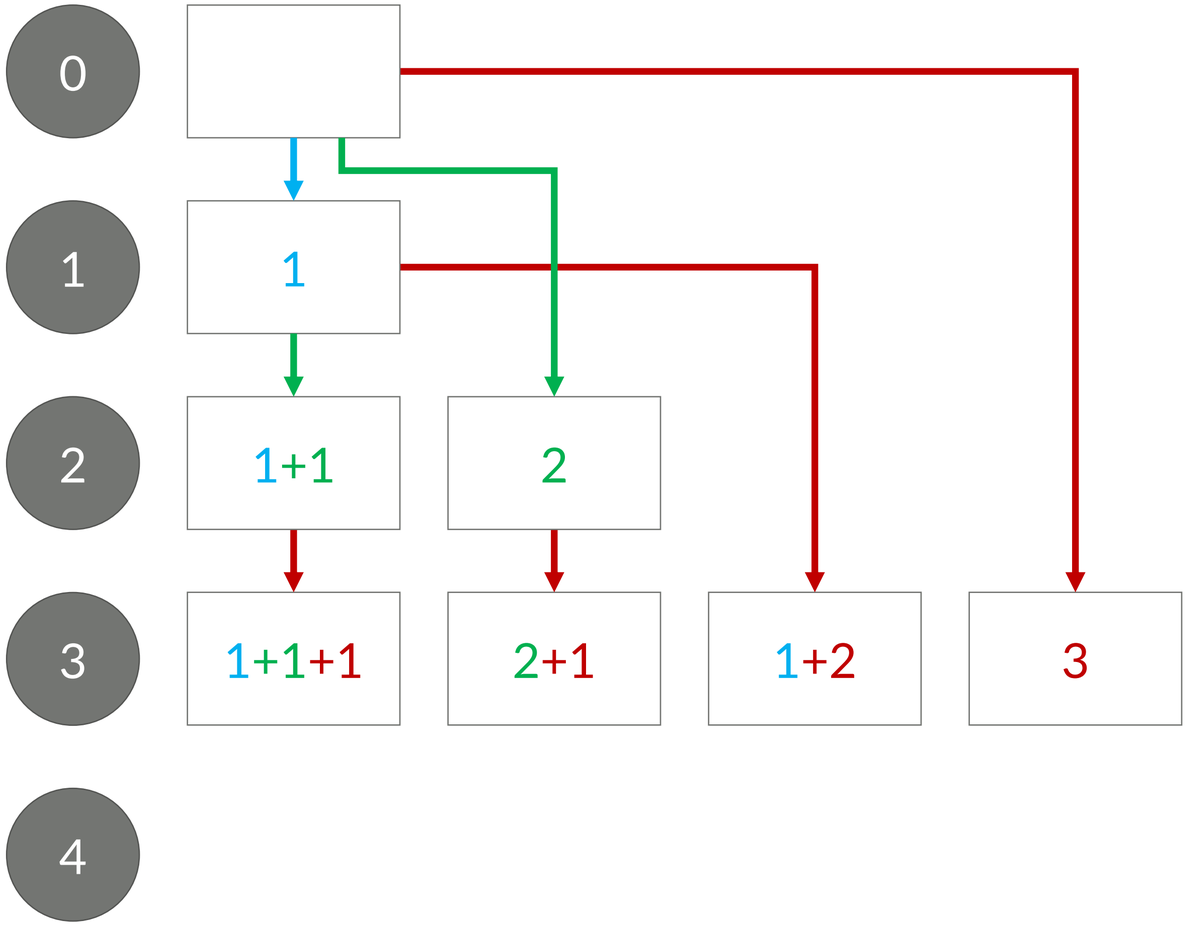
1, 2, 3 더하기

https://www.acmicpc.net/problem/9095



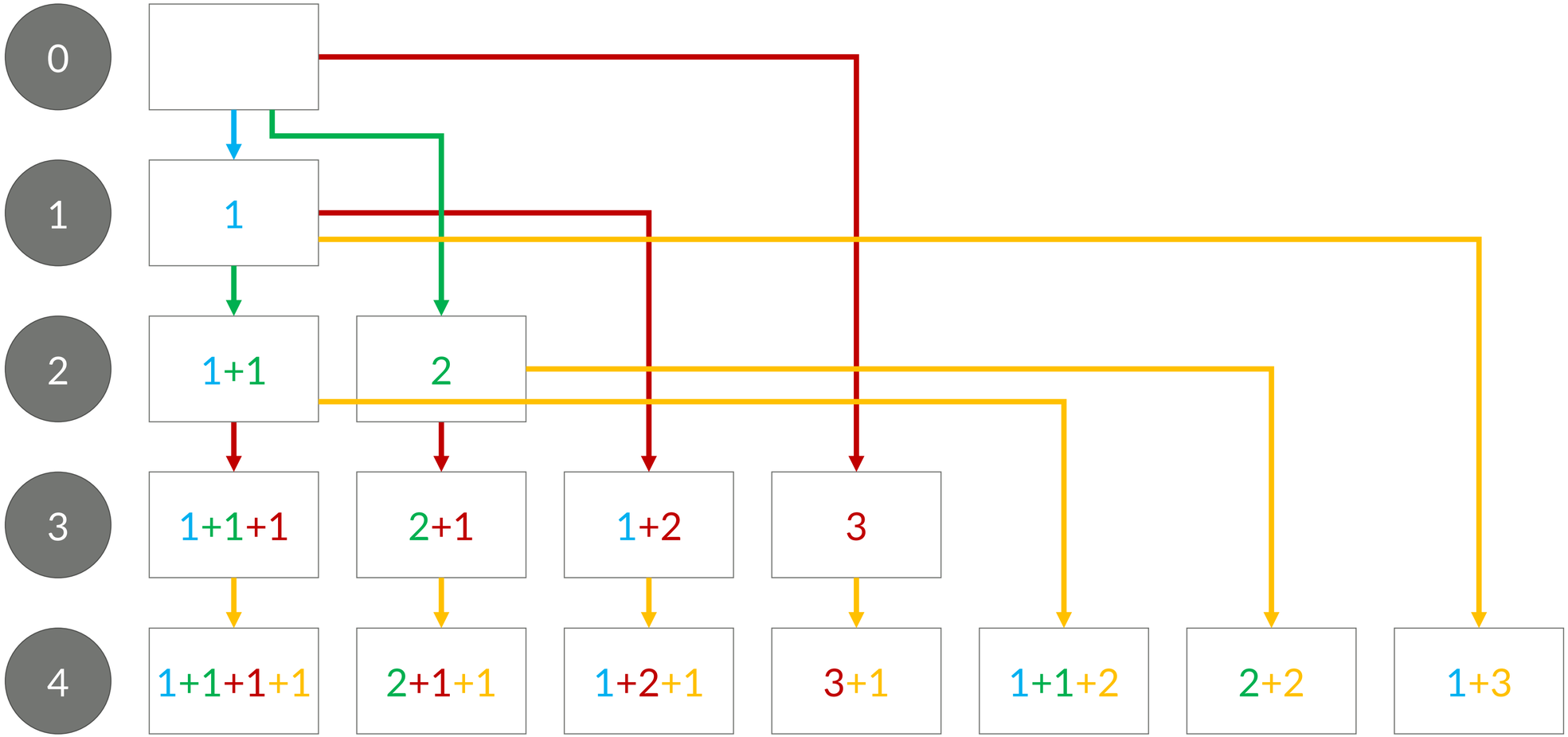
1, 2, 3 더하기

https://www.acmicpc.net/problem/9095



1, 2, 3 더하기

https://www.acmicpc.net/problem/9095



1, 2, 3 더하기 4

https://www.acmicpc.net/problem/15989

- 정수 n을 1, 2, 3의 합으로 나타내는 방법의 수를 구하는 문제
- 합을 이루고 있는 수의 순서만 다른 것은 같은 것으로 친다.
- n = 4
- 1+1+1+1
- 2+1+1 (1+1+2, 1+2+1)
- 2+2
- 1+3 (3+1)

1, 2, 3 더하기 4

https://www.acmicpc.net/problem/15989

0

1

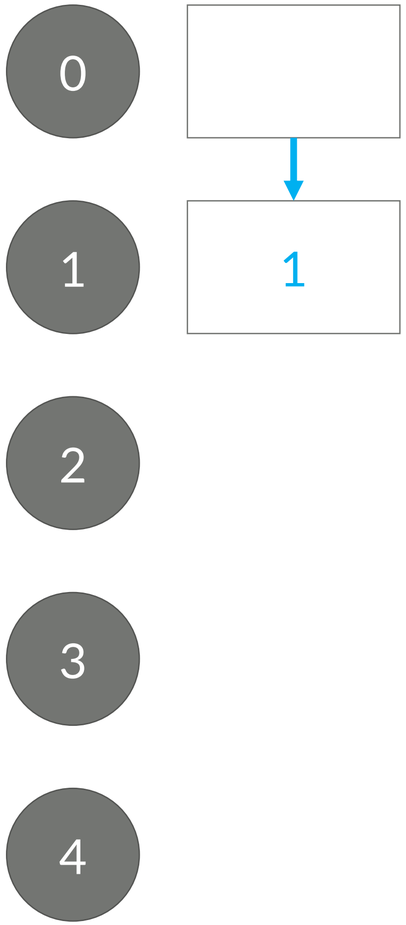
2

3

4

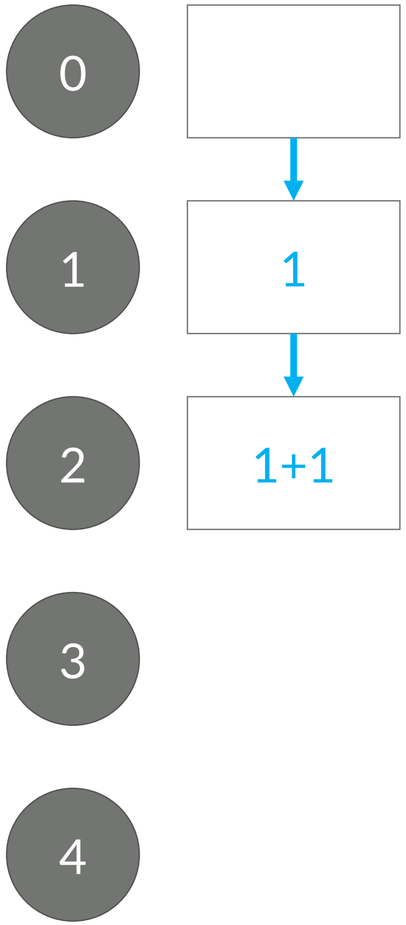
1, 2, 3 더하기 4

https://www.acmicpc.net/problem/15989



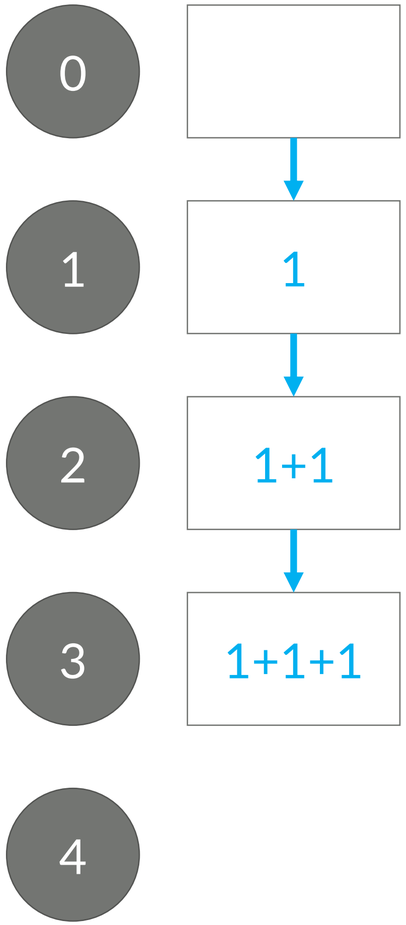
1, 2, 3 더하기 4

https://www.acmicpc.net/problem/15989



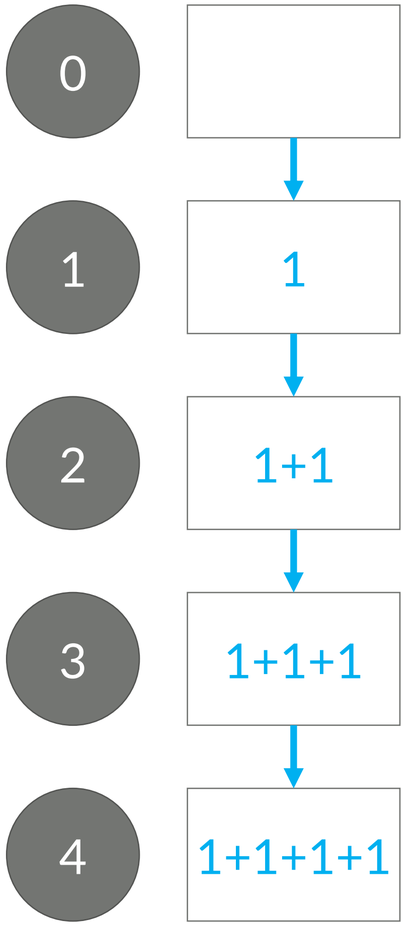
1, 2, 3 더하기 4

https://www.acmicpc.net/problem/15989



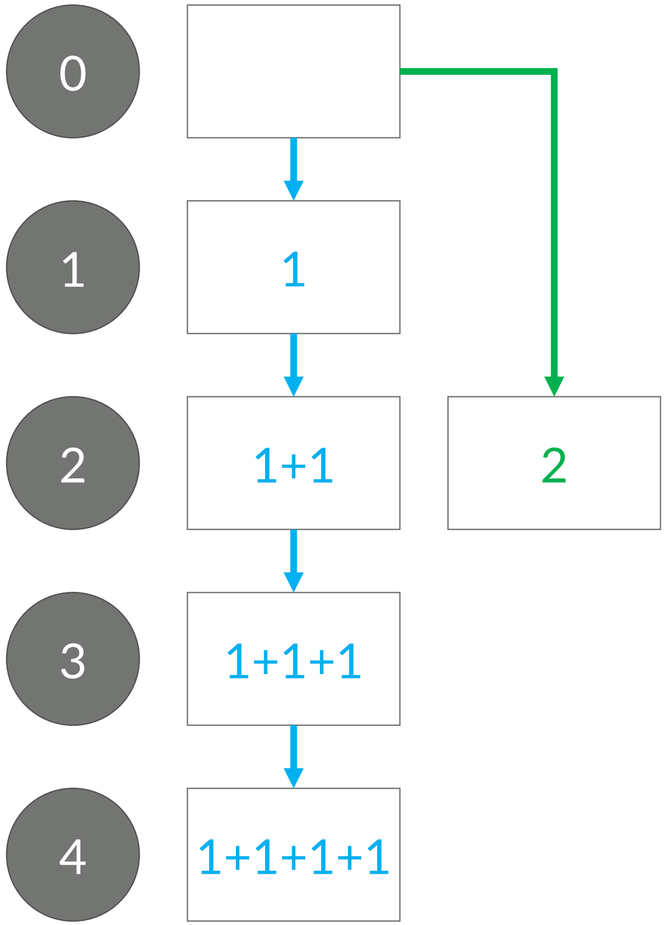
1, 2, 3 더하기 4

https://www.acmicpc.net/problem/15989



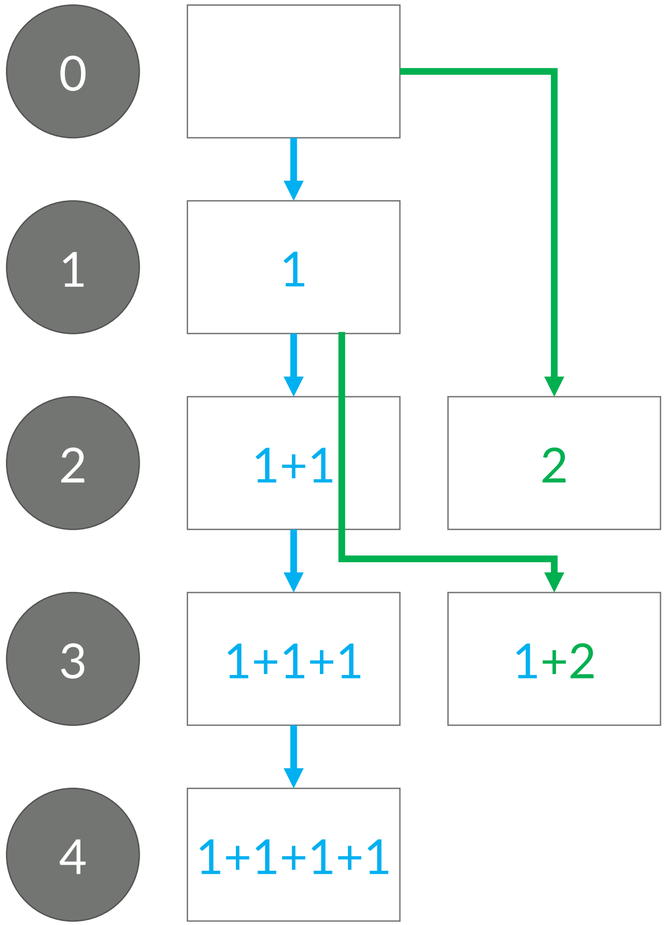
1, 2, 3 더하기 4

https://www.acmicpc.net/problem/15989



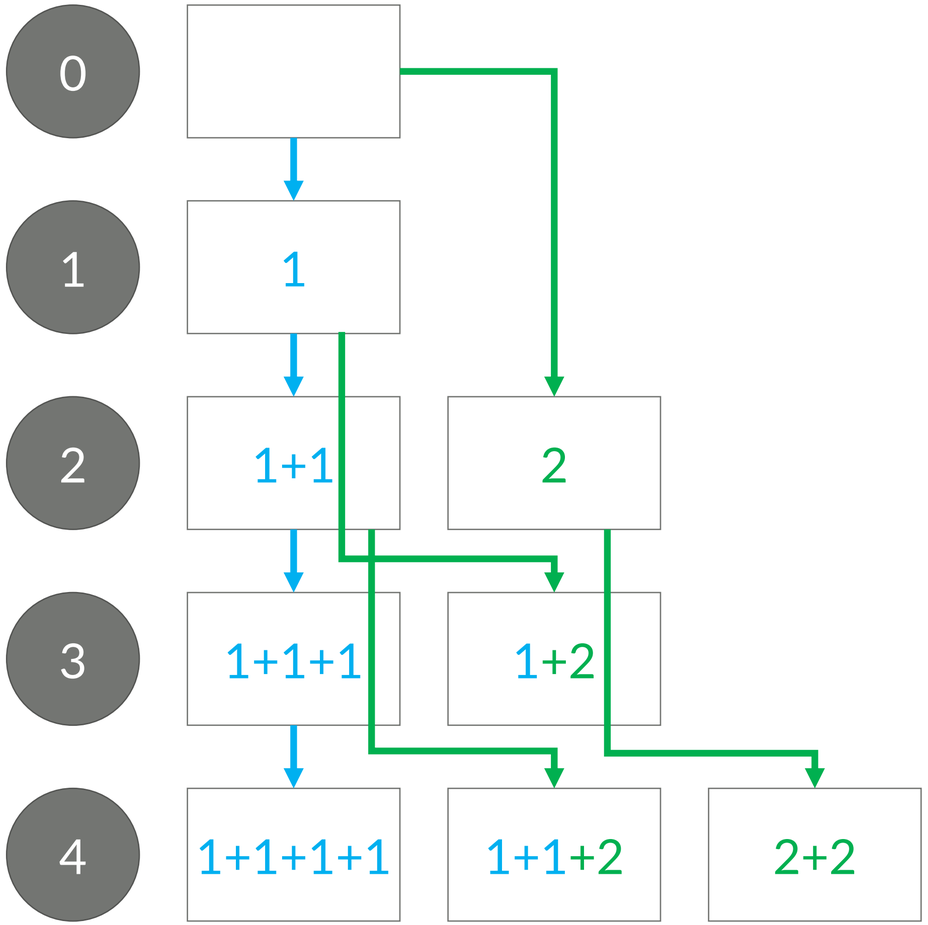
1, 2, 3 더하기 4

https://www.acmicpc.net/problem/15989



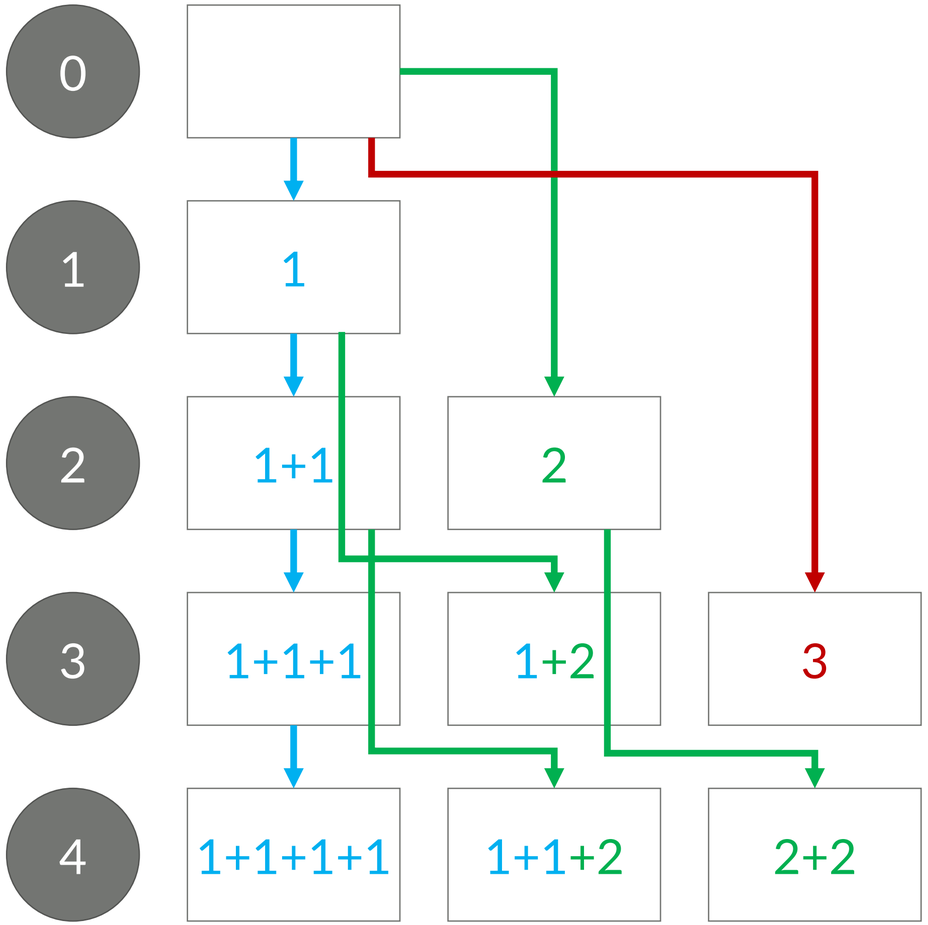
1, 2, 3 더하기 4

https://www.acmicpc.net/problem/15989



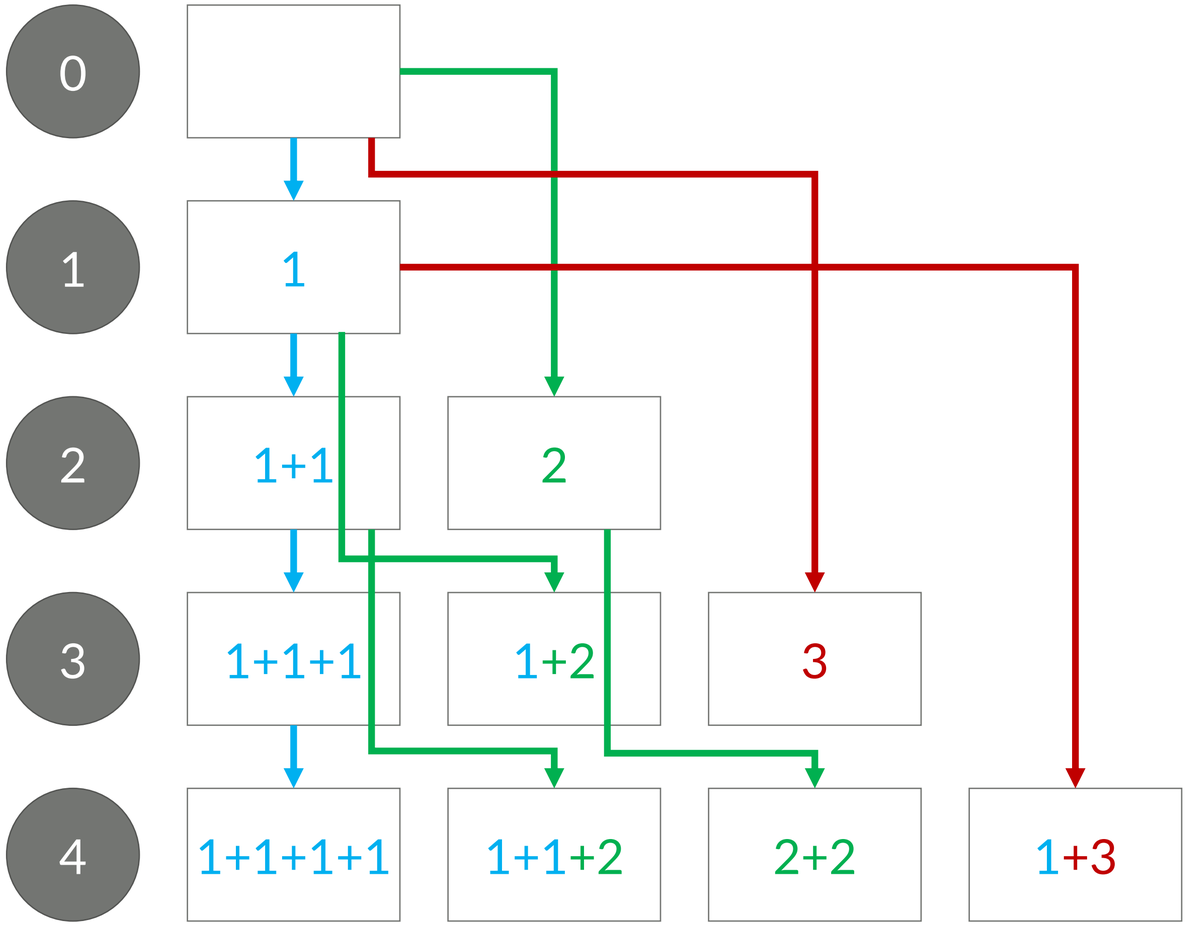
1, 2, 3 더하기 4

https://www.acmicpc.net/problem/15989



1, 2, 3 더하기 4

https://www.acmicpc.net/problem/15989



1, 2, 3 더하기

https://www.acmicpc.net/problem/9095

```
int n = 4;
int m = 3;
int nums = {1, 2, 3};
int d[n];
d[0] = 1;
for (int i=1; i<=n; i++) {
    for (int j=0; j<m; j++) {
        if (i-nums[j] >= 0) {
            d[i] += d[i-nums[j]];
        }
    }
}
```

1, 2, 3 더하기 4

https://www.acmicpc.net/problem/15989

```
int n = 4;
int m = 3;
int nums = {1, 2, 3};
int d[n];
d[0] = 1;
for (int j=0; j<m; j++) {
    for (int i=1; i<=n; i++) {
        if (i-nums[j] >= 0) {
            d[i] += d[i-nums[j]];
        }
    }
}
```

1, 2, 3 더하기 4

https://www.acmicpc.net/problem/15989

- 소스: http://codeplus.codes/ffed661143dc485eb0eaddc5a3f0f0c1

파일 합치기

https://www.acmicpc.net/problem/11066

- 각 장이 쓰여진 파일을 합쳐서 최종적으로 소설의 완성본이 들어있는 한 개의 파일을 만든
- 이 과정에서 두 개의 파일을 합쳐서 하나의 임시파일을 만들고, 이 임시파일이나 원래의 파일을 계속 두 개씩 합쳐서 소설의 여러 장들이 연속이 되도록 파일을 합쳐나가고, 최종적으로는 하나의 파일로 합친다
- 두 개의 파일을 합칠 때 필요한 비용(시간 등)이 두 파일 크기의 합이라고 가정할 때, 최종적인 한 개의 파일을 완성하는데 필요한 비용의 총 합

파일 합치기

https://www.acmicpc.net/problem/11066

- 파일의 크기가 40, 30, 30, 50인 경우 정답은 300이다.

파일 합치기

https://www.acmicpc.net/problem/11066

- 연속된 파일만 합칠 수 있다
- 파일은 2개의 연속된 파일을 합치는 것이다

파일 합치기

https://www.acmicpc.net/problem/11066

- 파일이 5개 있다고 하자. A_1 A_2 A_3 A_4 A_5
- 이렇게 5개의 파일을 합치는 방법은 4가지가 있다.
- (A_1) (A_2 A_3 A_4 A_5)
- (A_1 A_2) (A_3 A_4 A_5)
- (A_1 A_2 A_3) (A_4 A_5)
- (A_1 A_2 A_3 A_4) (A_5)

파일 합치기

https://www.acmicpc.net/problem/11066

- i번째 부터 j번째까지 파일이 있다고 하자. $A_i\ A_{i+1}\ \dots\ A_{j-1}\ A_j$
- 파일을 합치는 방법은 다음과 같이 나타낼 수 있다.
- $(A_i\ A_{i+1}\ \dots\ A_k)\ (A_{k+1}\ \dots\ A_{j-1}\ A_j)$

파일 합치기

https://www.acmicpc.net/problem/11066

- D[i][j] = i번째 부터 j번째까지 파일을 하나로 합치는 비용
- i번째 부터 j번째까지 파일이 있다고 하자. $A_i\ A_{i+1}\ \dots\ A_{j-1}\ A_j$
- 파일을 합치는 방법은 다음과 같이 나타낼 수 있다.
- $(A_i\ A_{i+1}\ \dots\ A_k)\ (A_{k+1}\ \dots\ A_{j-1}\ A_j)$
- D[i][k] + D[k+1][j] + 합치는 비용
- $i \leq k < j$

파일 합치기

https://www.acmicpc.net/problem/11066

- 소스: http://codeplus.codes/4ddc5731689e40df9ed22836129fc374

평범한 배낭

https://www.acmicpc.net/problem/12865

- N개의 물건이 있고, 각 물건은 무게 W[i]와 가치 V[i]를 갖는다.
- 배낭에는 무게 K까지만 물건을 넣을 수 있다.
- 배낭에 넣을 수 있는 물건 가치의 최댓값을 구하는 문제

평범한 배낭

https://www.acmicpc.net/problem/12865

- 각각의 물건을 배낭에 넣는 경우와 넣지 않는 경우가 있다.
- 그렇다면, 총 2^N 가지의 경우가 있다.
- $1 \leq N \leq 100$

평범한 배낭

https://www.acmicpc.net/problem/12865

- 배낭의 무게 $K \leq 100,000$ 이다.
- 각각의 무게에 대해서, 넣을 수 있는 가장 큰 가치를 알면 된다.

평범한 배낭

https://www.acmicpc.net/problem/12865

- D[i][j] = i번째 물건까지 고려했고, 배낭에 넣은 물건 무게의 합이 j일 때, 가치의 최댓값

평범한 배낭

https://www.acmicpc.net/problem/12865

- D[i][j] = i번째 물건까지 고려했고, 배낭에 넣은 물건 무게의 합이 j일 때, 가치의 최댓값
- i번째 물건을 가방에 넣지 않은 경우
- i번째 물건을 가방에 넣은 경우

평범한 배낭

https://www.acmicpc.net/problem/12865

- D[i][j] = i번째 물건까지 고려했고, 배낭에 넣은 물건 무게의 합이 j일 때, 가치의 최댓값
- i번째 물건을 가방에 넣지 않은 경우: D[i-1][j]
- i번째 물건을 가방에 넣은 경우

평범한 배낭

https://www.acmicpc.net/problem/12865

- D[i][j] = i번째 물건까지 고려했고, 배낭에 넣은 물건 무게의 합이 j일 때, 가치의 최댓값
- i번째 물건을 가방에 넣지 않은 경우: D[i-1][j]
- i번째 물건을 가방에 넣은 경우: D[i-1][j-w[i]] + v[i]

평범한 배낭

https://www.acmicpc.net/problem/12865

```
for (int i=1; i<=n; i++) {
    for (int j=1; j<=k; j++) {
        d[i][j] = d[i-1][j];
        if (j-w[i] >= 0) {
            d[i][j] = max(d[i][j], d[i-1][j-w[i]]+v[i]);
        }
    }
}
```

평범한 배낭

https://www.acmicpc.net/problem/12865

- 소스: http://codeplus.codes/9b8158eb68b748fa873848339f3dcceb

평범한 배낭

https://www.acmicpc.net/problem/12865

- D[i][j] = i번째 물건까지 고려했고, 배낭에 넣은 물건 무게의 합이 j일 때, 가치의 최댓값
- i번째 물건을 가방에 넣지 않은 경우: D[i-1][j]
- i번째 물건을 가방에 넣은 경우: D[i-1][j-w[i]] + v[i]
- D[i][j] = max(D[i-1][j], D[i-1][j-w[i]] + v[i]) 이기 때문에
- 1차원 다이나믹으로도 해결할 수 있다.

평범한 배낭

https://www.acmicpc.net/problem/12865

- D[i][j] = i번째 물건까지 고려했고, 배낭에 넣은 물건 무게의 합이 j일 때, 가치의 최댓값
- i번째 물건을 가방에 넣지 않은 경우: D[i-1][j]
- i번째 물건을 가방에 넣은 경우: D[i-1][j-w[i]] + v[i]
- D[i][j] = max(D[i-1][j], D[i-1][j-w[i]] + v[i]) 이기 때문에
- 1차원 다이나믹으로도 해결할 수 있다.
- D[j] = max(D[j], D[j-w[i]] + v[i])

평범한 배낭

https://www.acmicpc.net/problem/12865

```
for (int i=1; i<=n; i++) {
    for (int j=1; j<=k; j++) {
        if (j-w[i] >= 0) {
            d[j] = max(d[j], d[j-w[i]]+v[i]);
        }
    }
}
```

평범한 배낭

https://www.acmicpc.net/problem/12865

- 앞 페이지의 소스와 같은 방법은 물건을 중복해서 사용하게 된다.
- 중복 없이 배열을 채워야 한다.

평범한 배낭

https://www.acmicpc.net/problem/12865

- 앞 페이지의 소스와 같은 방법은 물건을 중복해서 사용하게 된다.
- 중복 없이 배열을 채워야 한다.
- 배열 채우는 순서를 뒤에서부터 채우면 된다.

평범한 배낭

https://www.acmicpc.net/problem/12865

```
for (int i=1; i<=n; i++) {
    for (int j=k; j>=1; j--) {
        if (j-w[i] >= 0) {
            d[j] = max(d[j], d[j-w[i]]+v[i]);
        }
    }
}
```

평범한 배낭

https://www.acmicpc.net/problem/12865

- 소스: http://codeplus.codes/58636fae2cef4c478a5d51f09c6f8025

기타리스트

https://www.acmicpc.net/problem/1495

- 첫 볼륨: S
- 연주해야 하는 곡의 개수 N개
- 가능한 볼륨의 범위: 0 ~ M

- i번 곡을 연주하기 전에 볼륨을 V[i]만큼 바꿔야 한다

- i번 곡을 연주하기 직전 볼륨이 P라면
- i번 곡은 P+V[i] 또는 P-V[i] 로 연주해야 한다
- 마지막 곡을 연주할 수 있는 볼륨 중 최대값

기타리스트

https://www.acmicpc.net/problem/1495

- D[i][j] = i번 곡을 볼륨 j로 연주할 수 있으면 1 없으면 0
- N = 3, S = 5, M = 10
- V[1] = 2, V[2] = 3, V[3] = 1

	0	1	2	3	4	5	6	7	8	9	10
0											
1											
2											
3											

기타리스트

https://www.acmicpc.net/problem/1495

- D[i][j] = i번 곡을 볼륨 j로 연주할 수 있으면 1 없으면 0
- N = 3, S = 5, M = 10
- V[1] = 2, V[2] = 3, V[3] = 1

	0	1	2	3	4	5	6	7	8	9	10
0						1					
1											
2											
3											

기타리스트

https://www.acmicpc.net/problem/1495

- D[i][j] = i번 곡을 볼륨 j로 연주할 수 있으면 1 없으면 0
- N = 3, S = 5, M = 10
- V[1] = 2, V[2] = 3, V[3] = 1

	0	1	2	3	4	5	6	7	8	9	10
0						1					
1				1				1			
2											
3											

기타리스트

https://www.acmicpc.net/problem/1495

- D[i][j] = i번 곡을 볼륨 j로 연주할 수 있으면 1 없으면 0
- N = 3, S = 5, M = 10
- V[1] = 2, V[2] = 3, V[3] = 1

	0	1	2	3	4	5	6	7	8	9	10
0						1					
1				1				1			
2	1				1		1				1
3											

기타리스트

https://www.acmicpc.net/problem/1495

- D[i][j] = i번 곡을 볼륨 j로 연주할 수 있으면 1 없으면 0
- N = 3, S = 5, M = 10
- V[1] = 2, V[2] = 3, V[3] = 1

	0	1	2	3	4	5	6	7	8	9	10
0						1					
1				1				1			
2	1				1		1				1
3		1		1		1		1		1	

기타리스트

https://www.acmicpc.net/problem/1495

- D[i][j] = i번 곡을 볼륨 j로 연주할 수 있으면 1 없으면 0
- D[i][j] 가 1이면
- D[i+1][j+V[i+1]]와 D[i+1][j-V[i+1]] 을 1로 만들 수 있다.

기타리스트

https://www.acmicpc.net/problem/1495

- 소스: http://codeplus.codes/0561e96ad9ae4e5baf0a2502338d62b4

뮤탈리스크

https://www.acmicpc.net/problem/12869

- SCV N개가 있다. ($1 \leq N \leq 3, 1 \leq$ 체력 ≤ 60)
- 뮤탈리스크 1개가 있다. 뮤탈리스크는 SCV를 공격한다.
- 한 번에 세 개의 SCV를 공격할 수 있다.
  - 첫 번째로 공격받는 SCV는 체력 9를 잃는다.
  - 두 번째로 공격받는 SCV는 체력 3을 잃는다.
  - 세 번째로 공격받는 SCV는 체력 1을 잃는다.
- SCV는 체력이 0 또는 그 이하가 되면 파괴되고, 한 번의 공격에서 같은 SCV를 여러 번 공격할 수 없다.
- SCV를 모두 파괴하기 위해 공격해야 하는 횟수의 최솟값을 구하는 문제

뮤탈리스크

https://www.acmicpc.net/problem/12869

- 한 번에 세 개의 SCV를 공격할 수 있다.
  - 첫 번째로 공격받는 SCV는 체력 9를 잃는다.
  - 두 번째로 공격받는 SCV는 체력 3을 잃는다.
  - 세 번째로 공격받는 SCV는 체력 1을 잃는다.
- 공격할 수 있는 순서는 총 6가지가 있다.
  - 1, 2, 3
  - 1, 3, 2
  - 2, 1, 3
  - 2, 3, 1
  - 3, 1, 2
  - 3, 2, 1

뮤탈리스크

https://www.acmicpc.net/problem/12869

- D[i][j][k] = SCV의 체력이 i, j, k 일 때, 모두 파괴하기 위해 공격해야 하는 횟수의 최솟값

뮤탈리스크

https://www.acmicpc.net/problem/12869

- D[i][j][k] = SCV의 체력이 i, j, k 일 때, 모두 파괴하기 위해 공격해야 하는 횟수의 최솟값
- 공격할 수 있는 순서는 총 6가지가 있다.
  - 1, 2, 3: D[i-9][j-3][k-1]
  - 1, 3, 2: D[i-9][j-1][k-3]
  - 2, 1, 3: D[i-3][j-9][k-1]
  - 2, 3, 1: D[i-1][j-9][k-3]
  - 3, 1, 2: D[i-3][j-1][k-9]
  - 3, 2, 1: D[i-1][j-3][k-9]

뮤탈리스크

https://www.acmicpc.net/problem/12869

- 소스: http://codeplus.codes/23a004fa03224651af56acc80242266b

괄호

https://www.acmicpc.net/problem/10422

- 길이 L이 주어졌을 때, 길이가 L인 올바른 괄호 문자열의 개수를 구하는 문제

괄호

https://www.acmicpc.net/problem/10422

- 길이 L이 주어졌을 때, 길이가 L인 올바른 괄호 문자열의 개수를 구하는 문제

올바른 괄호 문자열

괄호

https://www.acmicpc.net/problem/10422

- 길이 L이 주어졌을 때, 길이가 L인 올바른 괄호 문자열의 개수를 구하는 문제
- 첫 번째 글자와 대응하는 ')'는 어디에 있을까?

괄호

https://www.acmicpc.net/problem/10422

- 길이 L이 주어졌을 때, 길이가 L인 올바른 괄호 문자열의 개수를 구하는 문제
- 첫 번째 글자와 대응하는 ')'는 어디에 있을까? -> 알 수 없다

괄호

https://www.acmicpc.net/problem/10422

- 길이 L이 주어졌을 때, 길이가 L인 올바른 괄호 문자열의 개수를 구하는 문제
- 첫 번째 글자와 대응하는 ')'는 어디에 있을까? -> 알 수 없다 (i번째 글자)

괄호

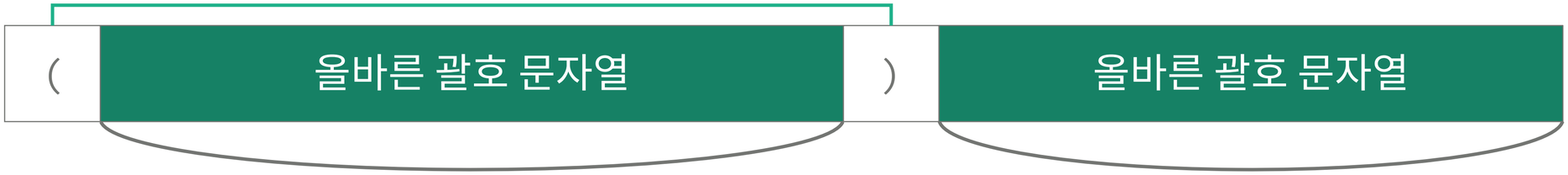
https://www.acmicpc.net/problem/10422

- 길이 L이 주어졌을 때, 길이가 L인 올바른 괄호 문자열의 개수를 구하는 문제
- 첫 번째 글자와 대응하는 ')'는 어디에 있을까? -> 알 수 없다

괄호

https://www.acmicpc.net/problem/10422

- 길이 L이 주어졌을 때, 길이가 L인 올바른 괄호 문자열의 개수를 구하는 문제
- 첫 번째 글자와 대응하는 ')'는 어디에 있을까? -> 알 수 없다



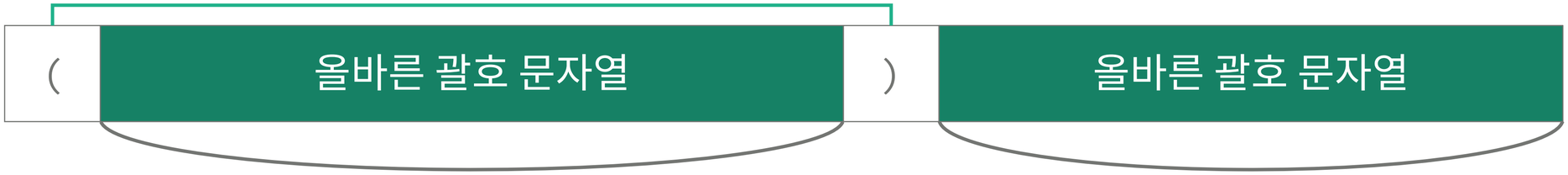
길이: i-2

길이: L-i

괄호

https://www.acmicpc.net/problem/10422

- 길이 L이 주어졌을 때, 길이가 L인 올바른 괄호 문자열의 개수를 구하는 문제
- 첫 번째 글자와 대응하는 ')'는 어디에 있을까? -> 알 수 없다



- D[L] = 길이가 L인 올바른 괄호 문자열의 개수
- D[L] = Σ(D[i-2] * D[L-i])

괄호

https://www.acmicpc.net/problem/10422

- 소스: http://codeplus.codes/d67285c09c89498cbbb67ea47a43f476

괄호

https://www.acmicpc.net/problem/10422

- 길이 N이 주어졌을 때, 길이가 N인 올바른 괄호 문자열의 개수를 구하는 문제
- D[N][L] = 길이가 N인 **괄호 문자열**, 짝이 맞지 않는 여는 괄호의 개수: L개
- 길이가 N인 올바른 괄호 문자열: D[N][0]

괄호

https://www.acmicpc.net/problem/10422

- 길이 N이 주어졌을 때, 길이가 N인 올바른 괄호 문자열의 개수를 구하는 문제
- D[N][L] = 길이가 N인 **괄호 문자열**, 짝이 맞지 않는 여는 괄호의 개수: L개

괄호

https://www.acmicpc.net/problem/10422

- 길이 N이 주어졌을 때, 길이가 N인 올바른 괄호 문자열의 개수를 구하는 문제
- D[N][L] = 길이가 N인 **괄호 문자열**, 짝이 맞지 않는 여는 괄호의 개수: L개

괄호

https://www.acmicpc.net/problem/10422

- 길이 N이 주어졌을 때, 길이가 N인 올바른 괄호 문자열의 개수를 구하는 문제
- D[N][L] = 길이가 N인 **괄호 문자열**, 짝이 맞지 않는 여는 괄호의 개수: L개
- $L < 0$이 되면 절대로 올바른 괄호 문자열을 만들 수 없기 때문에, $L \geq 0$에 대해서만 구한다

괄호

https://www.acmicpc.net/problem/10422

- 소스: http://codeplus.codes/22f03e47227748c3b74206bb0c0e86d0