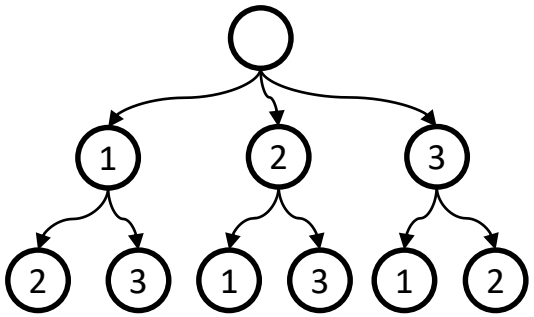
조합론의 완전검색(순열, 조합, 부분집합)

AD 보충수업 2일차

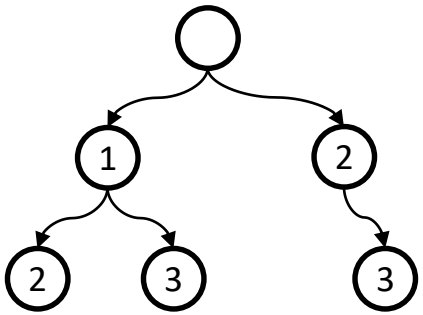
N = 3, R = 2, a = [1, 2, 3]

1 2
1 3
~~2 1~~
2 3
~~3 1~~
~~3 2~~



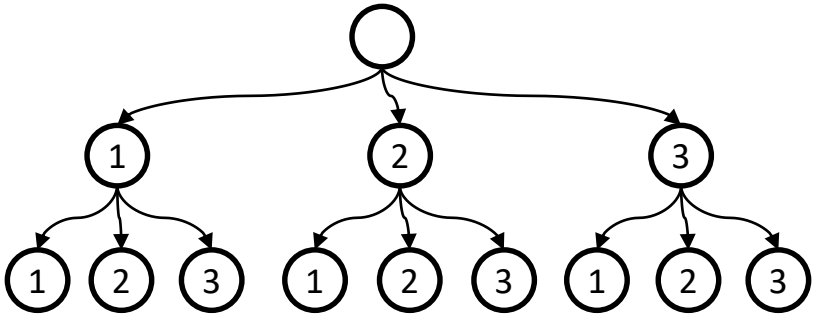
순열

1 2
1 3
2 3



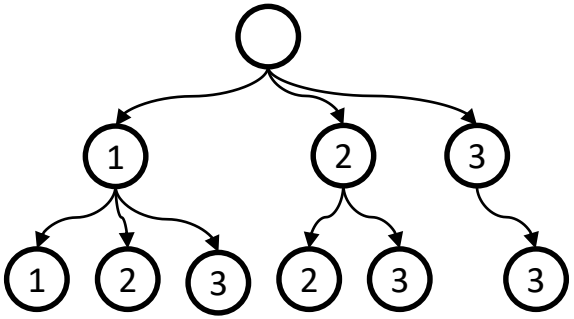
조합

1 1
1 2
1 3
~~2 1~~
2 2
2 3
~~3 1~~
~~3 2~~
3 3



중복 순열

1 1
1 2
1 3
2 2
2 3
3 3



중복 조합

- **순열 생성 재귀적 알고리즘1**

```
perm(n, r)
        if (r == 0) print_arr()
        else
                for (i : n – 1 ~ 0)
                        swap(a[i], a[n - 1])
                        t[r - 1] = a[n - 1]
                        perm(n - 1, r - 1)
                        swap(a[i], a[n - 1])
```

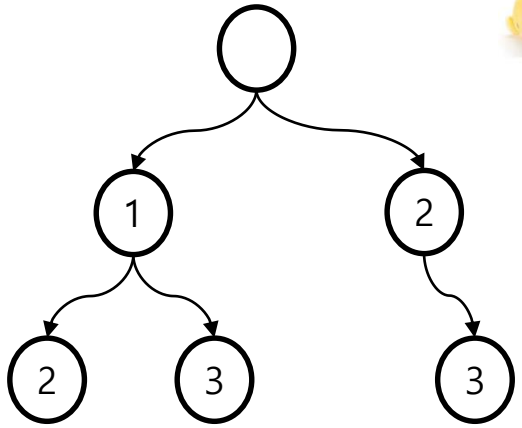
- **순열 생성 재귀적 알고리즘2**

```
perm(k)
        if (k == R) print_arr()
        else
                for (i : k ~ N - 1)
                        swap(k, i)
                        perm(k + 1)
                        swap(k, i)
```

- **순열 생성 재귀적 알고리즘2**

```
Visited[N-1]
perm(k)
        if (k == N) print_arr()
        else
                for (i : 0 ~ N - 1)
                        if (visited[i]) continue
                        t[k] = a[i]
                        visited[i] = true
                        perm(k + 1)
                        visited[i] = false
```

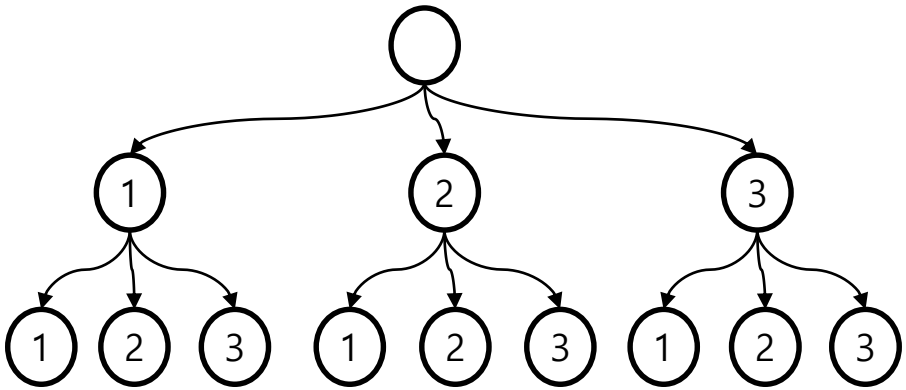
- **조합 생성 재귀적 알고리즘2**
  - 초기값 : k = 0, s = 0, N, R



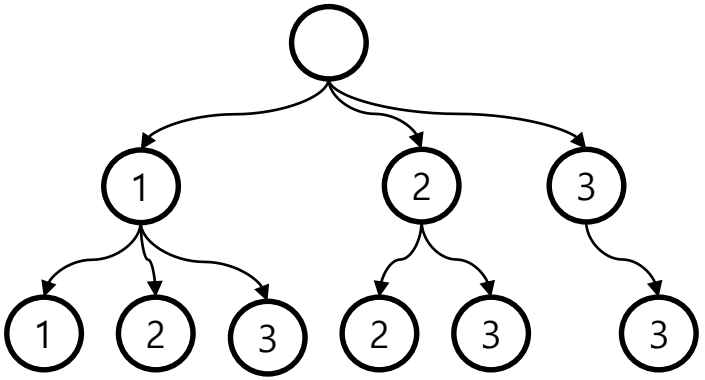

```
comb(k, s) // 깊이, 시작숫자
        if (k == R) print_arr()
        else
                for (int i : s ~ N - R + k )
                        t[k] = a[i]
                        comb(k + 1, i + 1)
```

- **중복 순열 생성 재귀적 알고리즘2** n^r

```
PI(k) // 깊이
        if (k == R) print_arr()
        else
                for (i : 0 ~ N - 1)
                        t[k] = a[i]
                        pi_r(k + 1)
```



- **중복조합 생성 재귀적 알고리즘2**
  - 초기값 : K = 0, s = 1, N, R




```
H(k, s) // 깊이, 시작숫자
        if (k == R) print_arr()
        else
                for (int i : s ~ N )
                        t[k] = a[i]
                        H(k + 1, i)
```

- **부분집합 생성 재귀적 알고리즘2**

```
def power_set_r(k):
    if k == N: print(a)
    else:
        a[k] = 1; power_set_r(k + 1)
        a[k] = 0; power_set_r(k + 1)


N = 3
a = [0] * N
power_set_r(0)
```

스타트와 링크

https://www.acmicpc.net/problem/14889

스타트와 링크

N = 6 일 경우

스타트 팀 0,1,2 그러면 링크 팀은 3,4,5

N//2개 뽑는 조합을 구하고

스타트 팀 = $S_{01} + S_{02} + S_{10} + S_{12} + S_{20} + S_{21}$

링크 팀 = $S_{34} + S_{35} + S_{43} + S_{45} + S_{53} + S_{54}$

구해진 조합에서 2개 나열하는 순열을 구한다.

스타트와 링크

```
N = int(input())
R = N // 2
t = [0] * R
mat = [list(map(int, input().split())) for _ in range(N)]

ans = 1e9
solve(0, 0)

print(ans)
```

0	1	2	3	4	5
1	0	2	3	4	5
1	2	0	3	4	5
1	2	3	0	4	5
1	2	3	4	0	5
1	2	3	4	5	0

스타트와 링크

```
def solve(k, s):
    global ans
    if k == R:
        …

    else:
        for i in range(s, N + (k - R) + 1):
            t[k] = i
            solve(k + 1, i + 1)
```

N이 6일 때 N//2 개의 조합을 구한다.

스타트와 링크

```
def solve(k, s):
    global ans
    if k == R:
        start = link = 0

        x = list(set([x for x in range(N)]) - set(t))

        for i in range(R - 1):
            for j in range(i + 1, R):
                start += (mat[t[i]][t[j]] + mat[t[j]][t[i]])
        for i in range(R - 1):
            for j in range(i + 1, R):
                link += (mat[x[i]][x[j]] + mat[x[j]][x[i]])

        ans = min(ans, abs(start - link))

    else:
        …
```

x = [3, 4, 5]

t = [0, 1, 2]

3개에서 2개 나열하는 순열

$S_{01} + S_{02} + S_{10} + S_{12} + S_{20} + S_{21}$

$S_{34} + S_{35} + S_{43} + S_{45} + S_{53} + S_{54}$

퇴사

https://www.acmicpc.net/problem/14501

퇴사

	1일	2일	3일	4일	5일	6일	7일
Ti	3	5	1	1	2	4	2
Pi	10	20	10	20	15	40	200

	1일	2일	3일	4일	5일	6일	7일
Ti	7	6	5	4	3	2	1
Pi	10	20	10	20	15	40	200

Ti 와 Pi 두 개의 인자가 있으며 항상 비례적이지는 않다.

	1일	2일	3일	4일	5일	6일	7일
Ti	1	1	1	1	1	1	1
Pi	10	20	10	20	15	40	200

완전검색 : 모든 부분 집합 조사

모든 부분 집합 ➔ Ti로 제외시키고 ➔ Pi로 최적해 구하기

퇴사

```
N = int(input())

Ti = [0] * N
Pi = [0] * N
Si = [0] * N

for i in range(N):
    Ti[i], Pi[i] = map(int, input().split())

ans = 0
solve(0)

print(ans)
```

	1일	2일	3일	4일	5일	6일	7일
Ti	3	5	1	1	2	4	2
Pi	10	20	10	20	15	40	200

퇴사

	1일	2일	3일	4일	5일	6일	7일
Ti	3	5	1	1	2	4	2
Pi	10	20	10	20	15	40	200

```
def solve(k):
    global ans
    if k == N:
        …
    else:
        Si[k] = 1; solve(k + 1)
        Si[k] = 0; solve(k + 1)
```

Si	1	1	1	1	1	1	1

N개에 대한 부분 집합

퇴사

	1일	2일	3일	4일	5일	6일	7일
Ti	3	5	1	1	2	4	2
Pi	10	20	10	20	15	40	200

선택유무조사

Si	1	1	1	1	1	1	1

```
def solve(k):
    global ans
    if k == N:
        for i in range(N):
            if Si[i]:
                for j in range(i + 1, i + Ti[i]):
                    if j >= N or Si[j] : return

        tsum = 0
        for i in range(N):
            if Si[i]:
                tsum += Pi[i]
        if tsum > ans : ans = tsum
    else:
        …
```

	1일	2일	3일	4일	5일	6일	7일
Ti	3	5	1	1	2	4	2
Pi	10	20	10	20	15	40	200

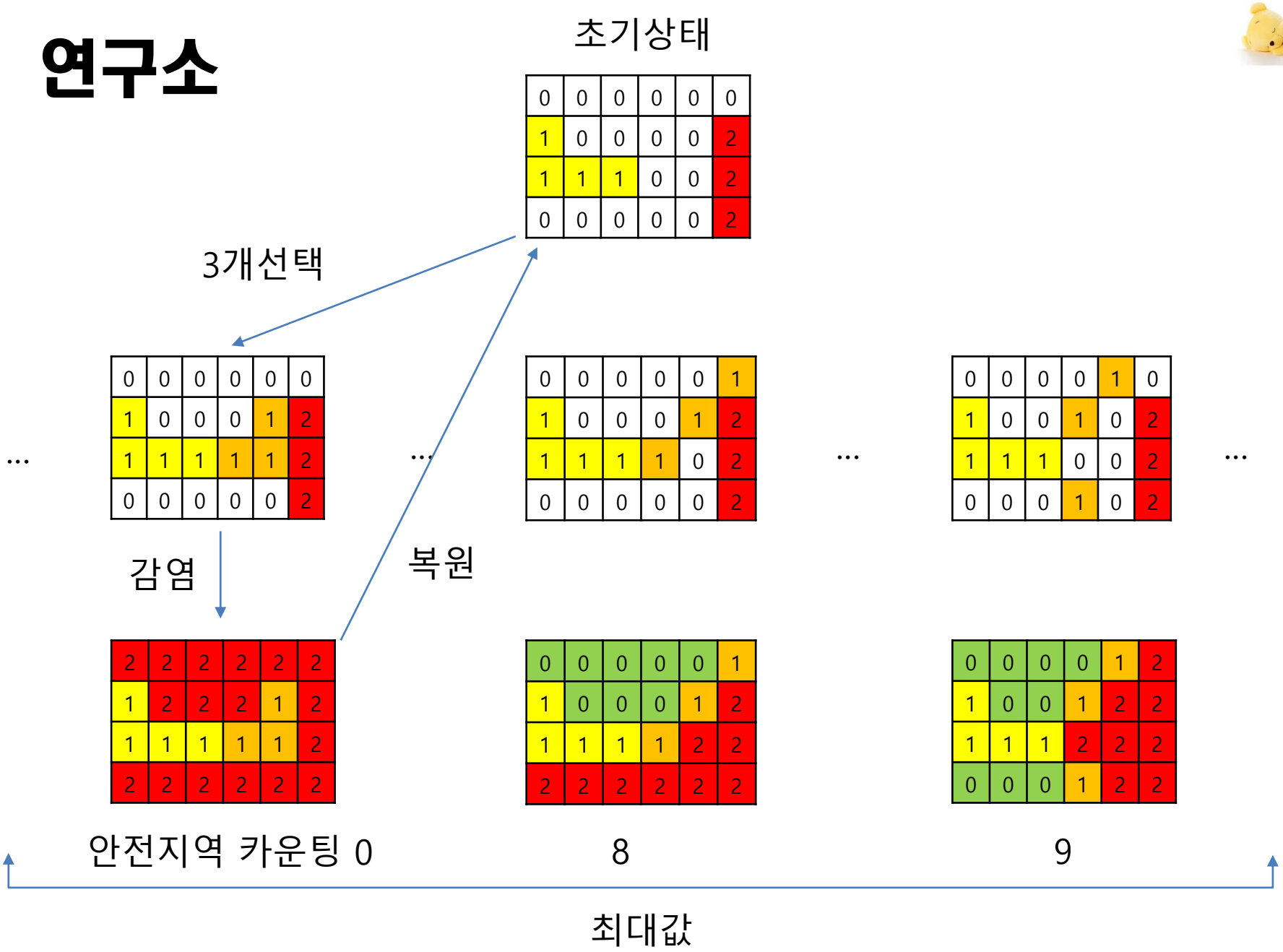
Si	1	0	0	0	1	0	0

10 + 15 = 25 가 최선인가?

연구소

https://www.acmicpc.net/problem/14502

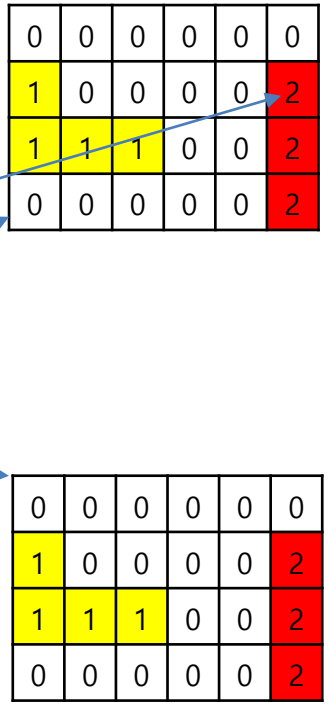
연구소



연구소

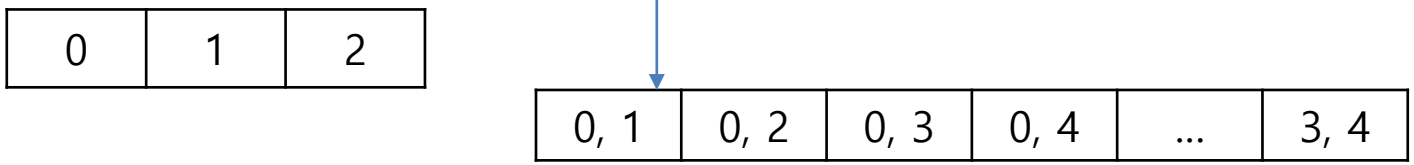
```
N, M = map(int, input().split())
mat = [list(map(int, input().split())) for _ in range(N)]

backup_mat = [[0] * M for i in range(N)]
virus_pos = []
safe_pos = []

for i in range(N):
    for j in range(M):
        if mat[i][j] == 2:
            virus_pos.append((i, j))
        elif mat[i][j] == 0:
            safe_pos.append((i, j))
        backup_mat[i][j] = mat[i][j]

ans = 0
combi = [0] * 3
solve(0, 0)
print(ans)
```

안전영역 개수에서 3개 선택 조합 저장할 배열



연구소

```
def solve(k, s):
    global ans
    if k == 3:
        ...
    else:
        for i in range(s, len(safe_pos) + (k - 3) + 1):
            combi[k] = i
            solve(k + 1, i + 1)
```



안전영역 개수에서 임의
3개 선택 ➔ 조합 생성

0	0	0	0	0	0
1	0	0	0	0	2
1	1	1	0	0	2
0	0	0	0	0	2

연구소

```
def solve(k, s):
    global ans
    if k == 3:
        for i in range(3):
            x, y = safe_pos[combi[i]]
            mat[x][y] = 1

        for x, y in virus_pos:
            virus_infact(x, y)

        ans = max(ans, sum(mat, []).count(0))

        for i in range(N):
            for j in range(M):
                mat[i][j] = backup_mat[i][j]
    else:
        ...
```

벽 세우기

0	0	0	0	0	0
1	0	0	0	1	2
1	1	1	1	1	2
0	0	0	0	0	2

감염시키기

2	2	2	2	2	2
1	2	2	2	1	2
1	1	1	1	1	2
2	2	2	2	2	2

안전영역세기

0	0	0	0	0	0
1	0	0	0	0	2
1	1	1	0	0	2
0	0	0	0	0	2

초기 상태로 복원

연구소

```
def virus_infact(x, y):
    mat[x][y] = 2
    for dx, dy in ((0, 1), (0, -1), (1, 0), (-1, 0)):
        xx, yy = x + dx, y + dy
        if not (0 <= xx < N and 0 <= yy < M): continue
        if not mat[xx][yy]:
            virus_infact(xx, yy)
```

0	0	0	0	0	0
1	0	0	0	1	2
1	1	1	1	1	2
0	0	0	0	0	2

(1, 5)에서 감염

2	2	2	2	2	2
1	2	2	2	1	2
1	1	1	1	1	2
0	0	0	0	0	2

치킨배달

https://www.acmicpc.net/problem/15686

치킨배달

https://www.acmicpc.net/problem/15686

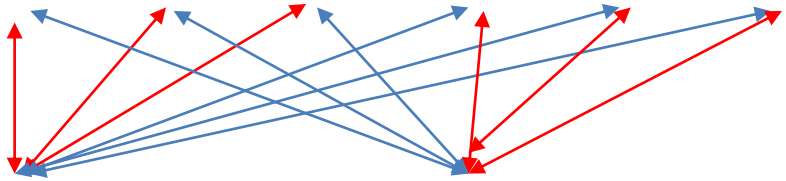
M = 2 (2개 치킨 집만 선택)

	0	1	2	3	4
0	0	2	0	1	0
1	1	0	1	0	0
2	0	0	0	0	0
3	2	0	0	1	1
4	2	2	0	1	2

집위치

0,3	1,0	1,2	3,3	3,4	4,3

모든 집과



0,1	3,0	4,0	4,1	4,4

치킨집위치

M개 선택한 치킨 집 사이의 거리 중 최소 거리 선택

치킨 집에서 임의 2개 선택

$_5C_2$

최소거리의 합 구하기

합 중 최소 구하기

치킨배달

	0	1	2	3	4
0	0	2	0	1	0
1	1	0	1	0	0
2	0	0	0	0	0
3	2	0	0	1	1
4	2	2	0	1	2

```
N, M = map(int, input().split())
mat = [list(map(int, input().split())) for _ in range(N)]

home, chicken = [], []
for i in range(N):
    for j in range(N):
        if mat[i][j] == 1:
            home.append((i, j))
        elif mat[i][j] == 2:
            chicken.append((i, j))

...

ans = 1e9
combi = [0] * M
solve(0, 0)
print(ans)
```

0,3	1,0	1,2	3,3	3,4	4,3

0,1	3,0	4,0	4,1	4,4

치킨 집에서 M개 선택하는
조합 생성

치킨배달

```
N, M = map(int, input().split())
mat = [list(map(int, input().split())) for _ in range(N)]

...

dist = [[0] * len(home) for i in range(len(chicken))]
for i in range(len(chicken)):
    for j in range(len(home)):
        dist[i][j] = abs(chicken[i][0] - home[j][0]) + abs(chicken[i][1] -
home[j][1])

ans = 1e9
combi = [0] * M
solve(0, 0)
print(ans)
```

모든 집과 모든 치킨 집 사이의 거리를 구해 놓기

치킨집 \ 집						
	2	2	2	5	6	6
	6	2	4	3	4	4
	7	3	5	4	5	3
	6	4	4	3	4	2
	5	7	5	2	1	1

치킨배달

집에서 최소거리 치킨집 선택

예

0	2

2	2	2	5	6	6
6	2	4	3	4	4
7	3	5	4	5	3
6	4	4	3	4	2
5	7	5	2	1	1

2+2+2+4+5+3=18

```
def solve(k, s):
    global ans
    if k == M:
        tsum = 0
        for h in range(len(home)):
            tmin = 1e9
            for c in combi:
                tmin = min(tmin, dist[c][h])
            tsum += tmin
        ans = min(ans, tsum)
    else:
        for i in range(s, len(chicken) + (k - M) + 1):
            combi[k] = i
            solve(k + 1, i + 1)
```

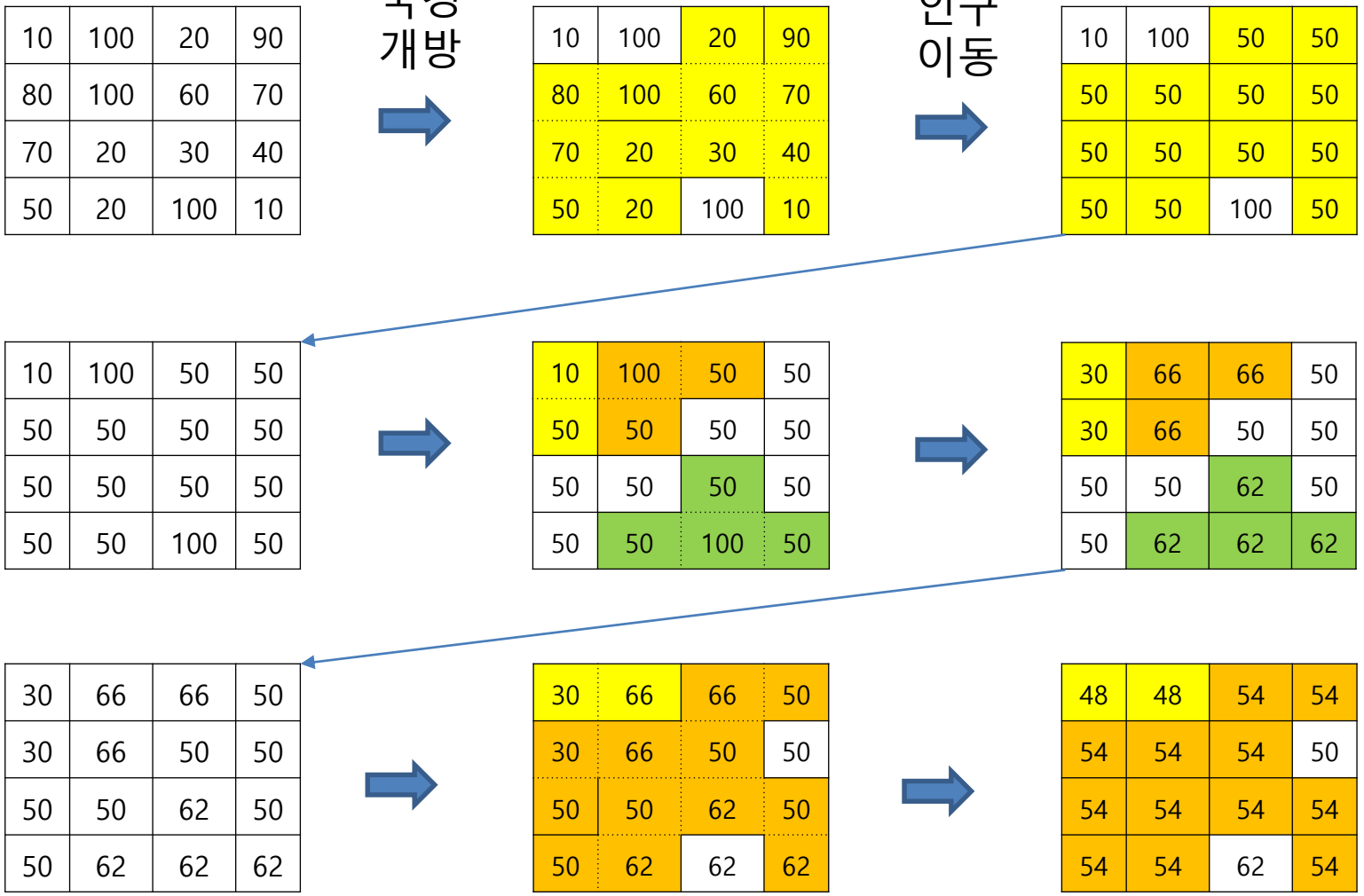
M길이 조합 구하기

인구이동

https://www.acmicpc.net/problem/16234

인구이동

L = 10, R = 50


국경
개방
인구
이동
10 100 20 90
80 100 60 70
70 20 30 40
50 20 100 10
10 100 20 90
80 100 60 70
70 20 30 40
50 20 100 10
10 100 50 50
50 50 50 50
50 50 50 50
50 50 100 50
10 100 50 50
50 50 50 50
50 50 50 50
50 50 100 50
30 66 66 50
30 66 50 50
50 50 62 50
50 62 62 62
30 66 66 50
30 66 50 50
50 50 62 50
50 62 62 62
48 48 54 54
54 54 54 50
54 54 54 54
54 54 62 54

인구이동

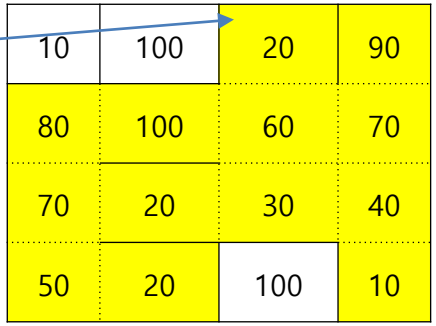
```
N, L, R = map(int, input().split())
mat = [list(map(int, input().split())) for _ in range(N)]

cnt = 0
while True:
    visited = [[0] * N for _ in range(N)]
    moved = False
    for i in range(N):
        for j in range(N):
            if not visited[i][j]:
                bfs(i, j)
    if moved: cnt += 1
    else: break

print(cnt)
```

매번 visited 새로 만들기

인구 이동이 없으면 중단

10	100	20	90
80	100	60	70
70	20	30	40
50	20	100	10

인구이동




```
def bfs(x, y):
    global moved
    q = []
    tList = []
    visited[x][y] = True
    q.append((x, y))

    while q:
        x, y = q.pop()
        tList.append((x, y))
        for dx, dy in ((0, 1), (0, -1), (1, 0), (-1, 0)):
            xx, yy = x + dx, y + dy
            if not (0 <= xx < N and 0 <= yy < N): continue
            if not visited[xx][yy] and L <= abs(mat[x][y] - mat[xx][yy]) <= R:
                visited[xx][yy] = 1
                q.append((xx, yy))

     ...
```

이동한 정점 저장

0,2	1,2	2,2	2,1	...	0,3

조건에 맞으면 이동

인구이동

```
def bfs(x, y):
    global moved
    q = []
    tList = []
    visited[x][y] = True
    q.append((x, y))

        ...

    tlen = len(tList)
    if tlen > 1:
        tsum = 0
        for x, y in tList:
            tsum += mat[x][y]
        for x, y in tList:
            mat[x][y] = tsum // tlen
        moved = True
```

인구 이동 있었음

10	100	20	90
80	100	60	70
70	20	30	40
50	20	100	10

10	100	50	50
50	50	50	50
50	50	50	50
50	50	100	50

0,2	1,2	2,2	2,1	...	0,3