### 파리퇴치

```
T = int(input())
for tc in range(1, T+1):
    N, M = map(int, input().split())
    arr = [list(map(int, input().split())) for _ in range(N)]
    result=0

    for i in range(N-M+1):
        for j in range(N-M+1):
            fly = 0
            for p in range(M):
                for q in range(M):
                    fly += arr[i+p][j+q]
                if result < fly:
                    result = fly

    print(f'#{tc} {result}')
```

### 풍선팡

```
di = [0, 1, 0, -1]
dj = [1, 0, -1, 0]

T = int(input())
for tc in range(1, T+1):
    N, M = map(int, input().split())
    arr = [list(map(int, input().split())) for _ in range(N)]
    result = 0

    for i in range(N):
        for j in range(M):
            flower = arr[i][j]
            for p in range(1, arr[i][j]+1):
                for k in range(4):
                    ni = i + di[k]*p
                    nj = j + dj[k]*p
                    if 0<=ni<N and 0<=nj<M:
                        flower += arr[ni][nj]
                if result < flower:
                    result = flower
    print(f'#{tc} {result}')
```

**파리퇴치3**

```
di = [0, 1, 0, -1]
dj = [1, 0, -1, 0]

T = int(input())
for tc in range(1, T+1):
    N, M = map(int, input().split())
    arr = [list(map(int, input().split())) for _ in range(N)]
    result = 0

    for i in range(N):
        for j in range(N):
            fly1 = arr[i][j]
            fly2 = arr[i][j]

            for p in range(1, M):
                for di, dj in [[1, 0],[0, 1],[0, -1],[-1, 0]]:
                    ni = i + di*p
                    nj = j + dj*p
                    if 0 <= ni < N and 0 <= nj < N:
                        fly1 += arr[ni][nj]

            for p in range(1, M):
                for di, dj in [[1, 1], [1, -1], [-1, 1], [-1, -1]]:
                    ni = i + di*p
                    nj = j + dj*p

                    if 0 <= ni < N and 0 <= nj < N:
                        fly2 += arr[ni][nj]

            if result < fly1:
                result = fly1
            if result < fly2:
                result = fly2

    print(f'#{tc} {result}')
```

### 풍선팡 보너스게임

```
di = [0, 1, 0, -1]
dj = [1, 0, -1, 0]

T = int(input())
for tc in range(1, T+1):
    N= int(input())
    arr = [list(map(int, input().split())) for _ in range(N)]
    result = 0

    for i in range(N):
        for j in range(N):
            flower = arr[i][j]
            for p in range(1, N):
                for k in range(4):
                    ni = i + di[k]*p
                    nj = j + dj[k]*p
                    if 0<=ni<N and 0<=nj<N:
                        flower += arr[ni][nj]
                if result < flower:
                    result = flower
    print(f'#{tc} {result}')
```

### 우주선착륙

```
di = [0, 1, 0, -1, -1, 1, 1, -1]
dj = [1, 0, -1, 0, 1, 1, -1, -1]

T = int(input())
for test_case in range(1, T+1):
    N, M = map(int, input().split())
    arr = [list(map(int, input().split())) for _ in range(N)]
    Huboji = 0

    for i in range(N):
        for j in range(M):
            photo_ok = 0
            for k in range(8):
                ni = i + di[k]
                nj = j + dj[k]
                if 0 <= ni < N and 0 <= nj < M:
                    if arr[i][j] > arr[ni][nj]:
                        photo_ok += 1
            if photo_ok >= 4:
                Huboji += 1

    print(f'#{test_case} {Huboji}')
```

### 고대유적

```
# 오른쪽, 아래로만 가는 방향
di = [0, 1]
dj = [1, 0]

T = int(input())
for test_case in range(1, T + 1):
    N, M = map(int, input().split())
    arr = [list(map(int, input().split())) for _ in range(N)]
    longest = 0  # 제일 긴 유적의 길이

    # arr을 순회해볼까
    for i in range(N):
        for j in range(M):
            if arr[i][j] == 1:  # arr[i][j]가 1이라고?
                for k in range(2):  # 그럼 아래방향, 오른쪽 방향으로만 탐색해보자
                    gili = 1    # gili = 현재탐색하는 유적의 길이
                                # 1을 찾은 순간 해당 유적의 길이는 최소 1이니까 현재
유적의 길이를 1로 설정
                    ni = i + di[k]
                    nj = j + dj[k]
                    while 0 <= ni < N and 0 <= nj < M and arr[ni][nj] == 1: # 범위
내에 있으면서, 다음 칸도 1이라면 반복하자
                        gili += 1   # 길이에 1을 더해!
                        ni = ni + di[k] # 한 칸을 더 나아가볼까
                        nj = nj + dj[k]


                    if longest < gili:  # 현재 유적 길이가 longest보다 길다면
                        longest = gili  # longest를 현재 유적 길이로 바꿔

    print(f'#{test_case} {longest}')
```

### 점점 커지는 당근의 개수

```
T = int(input())
for test_case in range(1, T + 1):
    N = int(input())
    arr = list(map(int, input().split()))
    max_carrot = 0
    carrots = 1

    for i in range(N - 1):
        if arr[i] < arr[i + 1]:
            carrots += 1
        else:
            carrots = 1

        if max_carrot < carrots:
                max_carrot = carrots

    print(f'#{test_case} {max_carrot}')
```

### 부분집합의 합

```
# N = 원소 개수, K = 원소의 합
# 집합 A의 부분 집합 중 N개의 원소를 갖고 있고, 원소의 합이 K인 부분집합의 개수를 구하시오.
T = int(input())

for test_case in range(1, T + 1):
    N, K = map(int, input().split())
    A = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
    n = len(A)
    count = 0

    for i in range(1 << n):   # i : 각 부분집합을 표현  1<<n: 부분집합의 개수
        num_sum = 0     # 원소의 합
        jip_sum = 0     # 원소의 개수

        for j in range(n):  # 원소의 수만큼 비트를 비교
            if i & (1 << j):  # i의 j번 비트가 1인 경우
                num_sum += A[j]  # A 리스트의 j번 원소 더하기
                jip_sum += 1     # 원소의 개수를 1 추가한다.

        if num_sum == K and jip_sum == N:   #원소의 합, 개수가 N, K랑 일치하면
            count += 1      # count를 1 증가시킨다.

    print(f'#{test_case} {count}')
```

**미로**

```
from collections import deque

# 4방향 좌표(상, 하, 좌, 우)
dx = [-1, 1, 0, 0]
dy = [0, 0, -1, 1]

def bfs(x, y):
    # 큐 생성
    queue = deque()
    # 시작위치를 큐에 추가
    queue.append((x, y))

    while queue:
        x, y = queue.popleft()

        # 현재 위치에서 네 방향으로의 위치 확인
        for i in range(4):
            nx = x + dx[i]
            ny = y + dy[i]

            # 주어진 범위 내에 있는지 확인
            if nx < 0 or nx >= N or ny < 0 or ny >= N:
                continue

            # 벽이면 무시하고 지나가라
            if arr[nx][ny] == 1:
                continue

            # 통로면 거기로 가자
            if arr[nx][ny] == 0:
                queue.append((nx, ny))
                arr[nx][ny] = 1

            # 도착점에 왔어? 1을 출력해
            if arr[nx][ny] == 3:
                return 1
    # 도착할 수 없어? 0을 출력해
    return 0

for test_case in range(10):
    case_num = int(input())
    N = 16
    arr = [list(map(int, input())) for _ in range(N)]
    start_x, start_y = 1, 1

    print(f'#{test_case+1} {bfs(start_x, start_y)}')
```

**회문**

```
T = int(input())
for tc in range(1, T+1):
    N, M = map(int, input().split()) # N은 문장 개수와 길이 / M은 부분적으로 조사할
길이
    arr = [input() for _ in range(N)]
    result = ''
# 가로 탐색
    for i in range(N):
        for j in range(N-M+1):
            if arr[i][j: j+M] == arr[i][j: j+M][::-1]:
                result = arr[i][j : j+M]
                break #
        else:
            continue
        break

# 세로 탐색
    for i in range(N-M+1):
        for j in range(N):
            vertical = ''
            for x in range(i, i+M):
                vertical += arr[x][j]
            if vertical == vertical[::-1]:
                result = vertical
                break
        else:
            continue
        break

    print(f'#{tc} {result}')
```