이진수

```
num_data = {'0' : '0000', '1' : '0001', '2' : '0010', '3':'0011',
            '4': '0100', '5': '0101', '6': '0110', '7': '0111',
            '8':'1000', '9':'1001', 'A':'1010', 'B':'1011',
            'C': '1100', 'D':'1101', 'E':'1110', 'F':'1111'}

T = int(input())
for test_case in range(1, T+1):
    N, arr = input().split()
    print(f'#{test_case} ', end='')
    for i in arr:
        if i in num_data:
            print(num_data[i], end = '')
    print()
```

이진수 표현

```
T = int(input())
for test_case in range(1, T+1):
    N, M = map(int, input().split())
    ans = 'OFF'

    if M & ((1 << N) - 1) == ((1 << N) - 1):
        ans = 'ON'

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

장훈이 선반 (DFS)

```
def dfs(cnt, sum_height):
    global ans

    # 기저조건
    # 기저조건 순서도 중요하다!

    # 1. 키의 합이 B 이상이면 종료
    # -> 현재까지 쌓은 탑의 높이 (sum_height)
    if sum_height >= B:
        # 제일 높이가 낮은 탑이 정답 :
        # 최소 탑의 높이는 재귀호출함수들이 "동시에" 사용함 -> 전역 변수로 빼야 된다.
        ans = min(ans, sum_height)
        return

    # 2. 모든 점원이 탑을 쌓는데 고려가 되었다면
    # -> 현재까지 쌓은 점원의 수 (cnt)
    if cnt == N:
        return

    # 재귀호출 파트
    # 쌓는다
    dfs(cnt + 1, sum_height + arr[cnt])
    # 안쌓는다
    dfs(cnt + 1, sum_height)



T = int(input())
for test_case in range(1, T+1):
    N, B = map(int, input().split())
    arr = list(map(int, input().split()))
    ans = int(1e9)

    dfs(0, 0)

    print(f'#{test_case} {abs(ans - B)}')
```

BFS

```
from collections import deque

def bfs(node):
    q = deque()
    q.append(node)
    visited[node] = 1
    print(node, end=' ')

    while q:
        now = q.popleft()
        for next in graph[now]:
            if visited[next] == 0:
                visited[next] = 1
                q.append(next)
                print(next, end=' ')

graph = [
    [],
    [2, 3, 8],
    [1, 7],
    [1, 4, 5],
    [3, 5],
    [3, 4],
    [7],
    [2, 6, 8],
    [1, 7]
]

visited = [0] * len(graph)

bfs(1)
```

빙산 (BFS)

```
def bfs(x, y):
    queue = deque
    queue.append(x, y)
    visited[x][y] = 0

    while queue:
        i, j = queue.popleft()

        for k in range(4):
            ni = i + di[k]
            nj = j + dj[k]
            # 범위내에 있고 방문하지 않았으면
            if 0 <= ni < N and 0 <= nj < M and visited[ni][nj] == 0:
                # 해당 위치가 빙산이면 방문표시 + 큐에 추가
                if arr[ni][nj] != 0:
                    visited[ni][nj] += 1
                    queue.append(ni, nj)

                # 해당 위치가 바닷물이라면
                if arr[ni][nj] == 0:
                    for k in range(4):
                        ni = i + di[k]
                        nj = j + dj[k]
                        # 범위 내에 있고 상하좌우 칸이 빙산이라면
                        if 0 <= ni < N and 0 <= nj < M and arr[ni][nj] == 1:
                            arr[ni][nj] -= 1    # 빙산의 숫자를 -1 하기

# 4방향 델타 설정
di = [0, 1, 0, -1]
dj = [1, 0, -1, 0]

N, M = map(int, input().split())    # N은 행, M은 열

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

visited = [[0] * M for _ in range(N)]
```

컴퓨터적 사고 문제

```
[문제]
n제곱이 3의 배수이면, n은 3의 배수이다.

[풀이]
대우: n이 3의 배수이면, n제곱이 3의 배수이다.
임의의 자연수 n은 n = 3k - 1, n = 3k - 2, n = 3k로 모두 표현이 가능하다.

이때 n = 3k라면,  n^2 = (3k)^2 = 9k^2 =  3 * (3k^2)이기에
n이 3의 배수이면,  n^2은 반드시 3의 배수임을 증명할 수 있다.

따라서 대우가 참이면, 명제도 참이기에 위 명제는 참이다.
```

```
[문제]
T(n) = T(n/2) + 1, T(1) = 1

T(n/2) = T(n/2^2) + 1

T(n) = T(n/2^2) + 1 + 1

T(n/2^2) = T(n/2^3) + 1

T(n) = T(n/2^3) + 1 + 1 + 1 = T(n/2^k) + k = T(1) + logn = logn + 1

-> O(T(n)) = O(logn)
```