25-1 이니로
알고리즘 멘토링

멘토 - 김수성

해킹 / 10282

백준 10282 / https://www.acmicpc.net/problem/10282

문제

최흉최악의 해커 yum3이 네트워크 시설의 한 컴퓨터를 해킹했다! 이제 서로에 의존하는 컴퓨터들은 점차 하나둘 전염되기 시작한다. 어떤 컴퓨터 a가 다른 컴퓨터 b에 의존한다면, b가 감염되면 그로부터 일정 시간 뒤 a도 감염되고 만다. 이때 b가 a를 의존하지 않는다면, a가 감염되더라도 b는 안전하다.

최흉최악의 해커 yum3이 해킹한 컴퓨터 번호와 각 의존성이 주어질 때, 해킹당한 컴퓨터까지 포함하여 총 몇 대의 컴퓨터가 감염되며 그에 걸리는 시간이 얼마인지 구하는 프로그램을 작성하시오.

입력

첫째 줄에 테스트 케이스의 개수가 주어진다. 테스트 케이스의 개수는 최대 100개이다. 각 테스트 케이스는 다음과 같이 이루어져 있다.

- 첫째 줄에 컴퓨터 개수 n, 의존성 개수 d, 해킹당한 컴퓨터의 번호 c가 주어진다(1 ≤ n ≤ 10,000, 1 ≤ d ≤ 100,000, 1 ≤ c ≤ n).
- 이어서 d개의 줄에 각 의존성을 나타내는 정수 a, b, s가 주어진다(1 ≤ a, b ≤ n, a ≠ b, 0 ≤ s ≤ 1,000). 이는 컴퓨터 a가 컴퓨터 b를 의존하며, 컴퓨터 b가 감염되면 s초 후 컴퓨터 a도 감염됨을 뜻한다.

각 테스트 케이스에서 같은 의존성 (a, b)가 두 번 이상 존재하지 않는다.

출력

각 테스트 케이스마다 한 줄에 걸쳐 총 감염되는 컴퓨터 수, 마지막 컴퓨터가 감염되기까지 걸리는 시간을 공백으로 구분지어 출력한다.

해킹 / 10282

첫 번째 테스트 케이스
2 -> 3 (5)
두 번째 테스트 케이스
1 -> 2 -> 3 (6)

예제 입력 1 복사

```
2
3 2 2
2 1 5
3 2 5
3 3 1
2 1 2
3 1 8
3 2 4
```

예제 출력 1 복사

```
2 5
3 6
```

해킹 / 10282

두 번째 테스트 케이스
1 -> 2 -> 3 (6)

컴퓨터 b가 바이러스에 걸리면 컴퓨터 a는
c의 시간 뒤에 바이러스에 걸림

각 컴퓨터를 정점으로 보면
b -> a로 가는 간선의 비용이 c로 정의 가능

해킹 / 10282

각 컴퓨터를 정점으로 보면
b -> a로 가는 간선의 비용이 c로 정의 가능

문제에서 주어진 의존성을 그래프로 변형

마지막 컴퓨터가 감염되기까지 걸리는 시간
-> 최단 거리의 최댓값

총 감염되는 컴퓨터의 수
-> 거리가 INF가 아닌 컴퓨터의 수

해킹 / 10282

C++

```cpp
const ll MAX = 10101;
const ll INF = 1e12;
ll n, m, k, d[MAX];
vector <pair<ll, ll>> adj[MAX];

using pll = pair<ll, ll>;
priority_queue <pll, vector<pll>, greater<pll>> pq;

void init(){
    while(!pq.empty()) pq.pop();
    for(int i = 1;i <= n;i++) adj[i].clear();
}
```

```cpp
int main(){
    ios::sync_with_stdio(0); // fastio
    cin.tie(0), cout.tie(0); // fastio

    ll t; cin >> t;
    while(t--) run();

    return 0;
}
```

```cpp
void run(){
    cin >> n >> m >> k; init();
    while(m--){
        ll s, e, c; cin >> s >> e >> c;
        adj[e].push_back({s, c});
    }

    for(int i = 0;i < MAX;i++) d[i] = INF;
    pq.push({0, k});

    while(!pq.empty()){
        auto[cd, cur] = pq.top(); pq.pop();
        if(d[cur] <= cd) continue;
        d[cur] = cd;

        for(auto& [nxt, co] : adj[cur]){
            if(d[nxt] <= cd + co) continue;
            pq.push({cd + co, nxt});
        }
    }

    ll cnt = 0, result = 0;
    for(int i = 1;i <= n;i++){
        // 바이러스에 감염되지 않았으면 건너 뜀
        if(d[i] == INF) continue;

        // 개수 증가 및 최댓값 갱신
        cnt++; result = max(result, d[i]);
    }

    cout << cnt << " " << result << "\n";
}
```

해킹 / 10282

Python

```python
def init():
    global pq, adj
    pq.clear()
    for i in range(1, n + 1):
        adj[i].clear()
```

```python
t = int(sys.stdin.readline())
d = [INF] * MAX
adj = [[] for _ in range(MAX)]
pq = []
for _ in range(t):
    run()
```

```python
def run():
    global n, m, k, d, adj, pq
    n, m, k = map(int, sys.stdin.readline().split())
    init()
    for _ in range(m):
        s, e, c = map(int, sys.stdin.readline().split())
        adj[e].append((s, c))

    for i in range(MAX):
        d[i] = INF
    heapq.heappush(pq, (0, k))

    while pq:
        cd, cur = heapq.heappop(pq)
        if d[cur] <= cd:
            continue
        d[cur] = cd

        for nxt, co in adj[cur]:
            if d[nxt] <= cd + co:
                continue
            heapq.heappush(pq, (cd + co, nxt))

    cnt = 0
    result = 0
    for i in range(1, n + 1):
        # 바이러스에 감염되지 않았으면 건너 뜀
        if d[i] == INF:
            continue

        # 개수 증가 및 최댓값 갱신
        cnt += 1
        result = max(result, d[i])
    print(cnt, result)
```

질문?

소가 길을 건너간 이유 7 / 14461

백준 14461 / https://www.acmicpc.net/problem/14461

문제

소가 길을 건너는 이유는 그냥 길이 많아서이다. 존의 농장에는 길이 너무 많아서, 길을 건너지 않고서는 별로 돌아다닐 수가 없다.

존의 농장에는 작은 정사각형 목초지가 N×N (3 ≤ N ≤ 100) 격자로 이루어져 있다. 농장의 바깥에는 높은 울타리가 있어서 소가 농장 밖으로 나갈 일은 없다. 이 농장에 사는 소 베시는 한 목초지에서 상하좌우로 인접한 다른 목초지로 이동할 수 있지만, 교통사고를 피하기 위해 차가 안 오는지 확인하고 길을 건너야 한다. 길을 건너는데는 T초 (0 ≤ T ≤ 1,000,000)가 걸린다.

존이 베시에게 체스 대결을 신청했다. 베시는 북서쪽 끝에 있는 목초지에서 남동쪽 끝에 있는 존의 집으로 가야 한다. 길이 멀기 때문에 베시는 가는 도중에 배가 고파진다. 그래서 길을 세 번 건널 때마다 목초지에 있는 풀을 먹어야 한다. 존의 집에 도착할 때도 해당되지만, 출발할 때는 해당되지 않는다. 목초지마다 풀이 자란 정도가 달라서, 풀을 먹는데 걸리는 시간도 다르다.

베시가 가능한 한 빨리 존의 집에 도착할 수 있도록 도와주자.

입력

첫 줄에 N과 T가 주어진다. 다음 N줄에는 목초지마다 풀을 먹는데 걸리는 시간이 N×N의 형태로 주어진다. 각각의 수는 모두 100,000 이하이다.

출력

베시가 존의 집까지 가는데 걸리는 최소 시간을 출력한다.

소가 길을 건너간 이유 7 / 14461

(0, 0)에서 (N – 1, N -1)까지 가는 최단 거리를 구해야 함
각 간선은 M의 비용이 들고, 3번의 이동마다
A[i][j]의 비용이 추가로 들음

(0,0) -> (0, 1) -> (0, 2) -> (0, 3) -> (1, 3) -> (2, 3) -> (2, 2)
-> (2, 3) -> (3, 3) = 2 * 8 + 10 + 5 = 31

예제 입력 1 복사

```
4 2
30 92 36 10
38 85 60 16
41 13 5 68
20 97 13 80
```

예제 출력 1 복사

```
31
```

소가 길을 건너간 이유 7 / 14461

최단 거리를 구해야 함 -> 다익스트라

하지만 각 간선을 단순히 M의 비용으로 연결하면
세 번째 이동의 추가 비용을 계산 할 수 없음

현재 몇 번째 이동인지 알아야 함
-> 추가적인 정보가 필요함
-> DP에서의 방법

소가 길을 건너간 이유 7 / 14461

현재 몇 번째 이동인지 알아야 함
-> 추가적인 정보가 필요함
-> DP에서의 방법

D[i][j] = (i, j)에서의 최단 거리
-> D[i][j][k] = (i, j)에서 k 번째 이동일 때의 최단 거리

소가 길을 건너간 이유 7 / 14461

D[i][j][k] = (i, j)에서 k 번째 이동일 때의 최단 거리

D[i][j][1] = D[py][px][0] + M
D[i][j][2] = D[py][px][1] + M
D[i][j][0] = D[py][px][2] + M + A[i][j]

소가 길을 건너간 이유 7 / 14461

C++

```cpp
const ll MAX = 3 * 10201;
const ll INF = 1e12;
ll n, m, d[MAX];
vector <pair<ll, ll>> adj[MAX];
ll a[101][101];
ll dx[4] = {0, 0, 1, -1}, dy[4] = {1, -1, 0, 0};

using pll = pair<ll, ll>;
priority_queue <pll, vector<pll>, greater<pll>> pq;

bool outrange(ll cy, ll cx){
    return cy <= 0 || cx <= 0 || cy > n || cx > n;
}

ll num(ll cy, ll cx, ll cnt){
    return cnt * 10101 + cy * 101 + cx;
}
```

```cpp
int main(){
    ios::sync_with_stdio(0); // fastio
    cin.tie(0), cout.tie(0); // fastio

    cin >> n >> m;
    for(int i = 1;i <= n;i++){
        for(int j = 1;j <= n;j++) cin >> a[i][j];
    }

    for(int i = 1;i <= n;i++){
        for(int j = 1;j <= n;j++){
            for(int k = 0;k < 4;k++){
                ll ny = i + dy[k], nx = j + dx[k];
                if(outrange(ny, nx)) continue;
                adj[num(i, j, 0)].push_back({num(ny, nx, 1), m});
                adj[num(i, j, 1)].push_back({num(ny, nx, 2), m});
                adj[num(i, j, 2)].push_back({num(ny, nx, 0), m + a[ny][nx]});
            }
        }
    }

    for(int i = 0;i < MAX;i++) d[i] = INF;
    pq.push({0, num(1, 1, 0)});

    while(!pq.empty()){
        auto[cd, cur] = pq.top(); pq.pop();
        if(d[cur] <= cd) continue;
        d[cur] = cd;

        for(auto& [nxt, co] : adj[cur]){
            if(d[nxt] <= cd + co) continue;
            pq.push({cd + co, nxt});
        }
    }

    ll result = INF;
    for(int i = 0;i < 3;i++) result = min(result, d[num(n, n, i)]);
    cout << result;

    return 0;
}
```

소가 길을 건너간 이유 7 / 14461

Python

```python
INF = 10**12
MAX = 3 * 10201

def outrange(cy, cx):
    return cy <= 0 or cx <= 0 or cy > n or cx > n

def num(cy, cx, cnt):
    return cnt * 10101 + cy * 101 + cx

n, m = map(int, input().split())
a = [[0] * (n + 2) for _ in range(n + 2)]
for i in range(1, n + 1):
    row = list(map(int, input().split()))
    for j in range(1, n + 1):
        a[i][j] = row[j - 1]

adj = [[] for _ in range(MAX)]
d = [INF] * MAX
dx = [0, 0, 1, -1]
dy = [1, -1, 0, 0]
```

```python
for i in range(1, n + 1):
    for j in range(1, n + 1):
        for k in range(4):
            ny = i + dy[k]
            nx = j + dx[k]
            if outrange(ny, nx):
                continue
            adj[num(i, j, 0)].append((num(ny, nx, 1), m))
            adj[num(i, j, 1)].append((num(ny, nx, 2), m))
            adj[num(i, j, 2)].append((num(ny, nx, 0), m + a[ny][nx]))

pq = []
heapq.heappush(pq, (0, num(1, 1, 0)))

while pq:
    cd, cur = heapq.heappop(pq)
    if d[cur] <= cd:
        continue
    d[cur] = cd
    for nxt, co in adj[cur]:
        if d[nxt] <= cd + co:
            continue
        heapq.heappush(pq, (cd + co, nxt))

result = INF
for i in range(3):
    result = min(result, d[num(n, n, i)])
print(result)
```

질문?

수열과 개구리 / 32294

백준 32294 / https://www.acmicpc.net/problem/32294

문제

길이가 n이고 양의 정수로 구성된 수열 a와 b가 있습니다. 두 수열의 인덱스는 1부터 시작합니다.

이 수열 위에 개구리 한 마리가 있습니다. 개구리는 초기에 어떤 정수 위치 $1 \le x \le n$에서 출발하며, 자신의 위치가 수열 밖[†]이 될 때까지 다음을 반복합니다.

b_x초 동안 기다린 뒤, 위치 $x - a_x$로 이동하거나 위치 $x + a_x$로 이동합니다. 이때 이동한 위치가 x의 새로운 값이 됩니다.

개구리가 시작하는 위치 x에 대해, 개구리의 위치가 수열 밖이 될 수 있는 최초의 시각을 $f(x)$초라고 정의할 때, 여러분은 $f(1), f(2), \cdots, f(n)$의 값을 모두 구해야 합니다.

† 어떤 위치 x에 대해서, $1 \le x \le n$이면 **수열 안**, $x < 1$ 또는 $x > n$이면 **수열 밖**이라고 부릅니다.

입력

첫 번째 줄에 두 수열의 길이 n이 주어집니다. ($1 \le n \le 2 \cdot 10^5$)

두 번째 줄에 n개의 정수 $a_1, a_2, \ldots, a_n$이 공백으로 구분되어 주어집니다. ($1 \le a_i \le n$)

세 번째 줄에 n개의 정수 $b_1, b_2, \ldots, b_n$이 공백으로 구분되어 주어집니다. ($1 \le b_i \le 10^6$)

출력

한 줄에 $f(1), f(2), \cdots, f(n)$의 값을 공백으로 구분하여 순서대로 출력합니다.

문제의 제한에 따라 모든 $1 \le i \le n$에 대해 $f(i)$가 유한함을 증명할 수 있습니다.

수열과 개구리 / 32294

각 인덱스 i에 대해서 i - A[i] 또는 i + A[i]로 이동 할 때
B[i]의 시간이 들음

각 인덱스 i에 대해서 0 이하 또는 N 초과로 이동 할 때
최단거리 출력

예제 입력 1 복사

```
5
3 4 2 5 1
2 5 1 4 3
```

예제 출력 1 복사

```
2 5 3 4 3
```

수열과 개구리 / 32294

각 인덱스 i에 대해서 i - A[i] 또는 i + A[i]로 이동 할 때
B[i]의 시간이 들음

각 인덱스 i는 i + A[i] 또는 i - A[i]와 연결되어 있고
비용이 B[i]인 간선을 가지는 그래프로 모델링 할 수 있음

수열과 개구리 / 32294

각 인덱스 i에 대해서 다익스트라를 돌리고
x <= 0 || x > n 인덱스 x의 최단거리를 구하면 됨
B[i]가 1e6까지 들어오기 때문에 모든 x를 시도 할 수는 없음

x <= 0 || x > n 일 때 간선을 0과 연결하면 d[0]의 값이 정답이 됨
-> O(N ^ 2 log N) N이 2e5이므로 시간 초과

수열과 개구리 / 32294

각 인덱스 i에 대해서 i -> 0을 구해야 함
저번주에 풀었던 파티 문제를 생각해보자

각 인덱스 i에 대해서 i -> 0를 구하는 건 시간이 많이 들음

모든 간선을 뒤집고 0 -> i 를 구하면 다익스트라를
한번만 사용해서 구할 수 있음

수열과 개구리 / 32294

C++

```cpp
const ll MAX = 201010;
const ll INF = 1e12;
ll n, m, d[MAX];
vector <pair<ll, ll>> adj[MAX];
ll a[MAX], b[MAX];

using pll = pair<ll, ll>;
priority_queue <pll, vector<pll>, greater<pll>> pq;
```

```cpp
int main(){
    ios::sync_with_stdio(0); // fastio
    cin.tie(0), cout.tie(0); // fastio

    cin >> n;
    for(int i = 1;i <= n;i++) cin >> a[i];
    for(int i = 1;i <= n;i++) cin >> b[i];

    for(int i = 1;i <= n;i++){
        ll nx = i - a[i];
        if(nx <= 0) adj[0].push_back({i, b[i]});
        else adj[nx].push_back({i, b[i]});

        nx = i + a[i];
        if(nx > n) adj[0].push_back({i, b[i]});
        else adj[nx].push_back({i, b[i]});
    }


    for(int i = 0;i < MAX;i++) d[i] = INF;
    pq.push({0, 0});

    while(!pq.empty()){
        auto[cd, cur] = pq.top(); pq.pop();
        if(d[cur] <= cd) continue;
        d[cur] = cd;

        for(auto& [nxt, co] : adj[cur]){
            if(d[nxt] <= cd + co) continue;
            pq.push({cd + co, nxt});
        }
    }

    for(int i = 1;i <= n;i++) cout << d[i] << " ";

    return 0;
}
```

수열과 개구리 / 32294

Python

```python
MAX = 201010
INF = 10**12

n = int(input())
a = [0] + list(map(int, input().split()))
b = [0] + list(map(int, input().split()))

adj = [[] for _ in range(MAX)]
d = [INF] * MAX
pq = []

for i in range(1, n + 1):
    nx = i - a[i]
    if nx <= 0:
        adj[0].append((i, b[i]))
    else:
        adj[nx].append((i, b[i]))

    nx = i + a[i]
    if nx > n:
        adj[0].append((i, b[i]))
    else:
        adj[nx].append((i, b[i]))
```

```python
heapq.heappush(pq, (0, 0))

while pq:
    cd, cur = heapq.heappop(pq)
    if d[cur] <= cd:
        continue
    d[cur] = cd

    for nxt, co in adj[cur]:
        if d[nxt] <= cd + co:
            continue
        heapq.heappush(pq, (cd + co, nxt))

print(' '.join(str(d[i]) for i in range(1, n + 1)))
```

질문?

4주차 - 분할 정복

분할 정복

큰 문제를 바로 해결 할 수 있을 때까지 분할 후 (분할)
하위 문제의 해답을 합치면서 큰 문제의 답을 구하는 알고리즘 (정복)

큰 문제를 분할하는 법과 작은 문제를 합치는 방법이 같아야 함
-> 재귀함수로 구현

종이의 개수 / 1780

백준 1780 / https://www.acmicpc.net/problem/1780

문제

N×N크기의 행렬로 표현되는 종이가 있다. 종이의 각 칸에는 -1, 0, 1 중 하나가 저장되어 있다. 우리는 이 행렬을 다음과 같은 규칙에 따라 적절한 크기로 자르려고 한다.

1. 만약 종이가 모두 같은 수로 되어 있다면 이 종이를 그대로 사용한다.
2. (1)이 아닌 경우에는 종이를 같은 크기의 종이 9개로 자르고, 각각의 잘린 종이에 대해서 (1)의 과정을 반복한다.

이와 같이 종이를 잘랐을 때, -1로만 채워진 종이의 개수, 0으로만 채워진 종이의 개수, 1로만 채워진 종이의 개수를 구해내는 프로그램을 작성하시오.

입력

첫째 줄에 N($1 \le N \le 3^7$, N은 3^k 꼴)이 주어진다. 다음 N개의 줄에는 N개의 정수로 행렬이 주어진다.

출력

첫째 줄에 -1로만 채워진 종이의 개수를, 둘째 줄에 0으로만 채워진 종이의 개수를, 셋째 줄에 1로만 채워진 종이의 개수를 출력한다.

종이의 개수 / 1780

예제 입력 1 복사

```
9
0 0 0 1 1 1 -1 -1 -1
0 0 0 1 1 1 -1 -1 -1
0 0 0 1 1 1 -1 -1 -1
1 1 1 0 0 0 0 0 0
1 1 1 0 0 0 0 0 0
1 1 1 0 0 0 0 0 0
0 1 -1 0 1 -1 0 1 -1
0 -1 1 0 1 -1 0 1 -1
0 1 -1 1 0 -1 0 1 -1
```

예제 출력 1 복사

```
10
12
11
```

종이의 개수 / 1780

1. 만약 종이가 모두 같은 수로 되어 있다면 이 종이를 그대로 사용한다.
2. (1)이 아닌 경우에는 종이를 같은 크기의 종이 9개로 자르고, 각각의 잘린 종이에 대해서 (1)의 과정을 반복한다.

분할 – 종이가 모두 같은 수로 되어 있지 않으면 9개로 분할

정복 – 종이가 모두 같은 수로 되어 있으면 종이의 개수 증가

```
// {sx, sy}, {ex, ey}의 종이를 볼 때 -1, 0, 1로 채워진 종이의 개수
tuple <ll, ll, ll> dnc(ll sx, ll sy, ll ex, ll ey){

}
```

종이의 개수 / 1780

1. 만약 종이가 모두 같은 수로 되어 있다면 이 종이를 그대로 사용한다.
2. (1)이 아닌 경우에는 종이를 같은 크기의 종이 9개로 자르고, 각각의 잘린 종이에 대해서 (1)의 과정을 반복한다.

분할 – 종이가 모두 같은 수로 되어 있지 않으면 9개로 분할

```
// {sx, sy}, {ex, ey}의 종이를 볼 때 -1, 0, 1로 채워진 종이의 개수
tuple <ll, ll, ll> dnc(ll sx, ll sy, ll ex, ll ey){
    bool flag = 1; ll num = a[sy][sx];
    for(int i = sy;i <= ey;i++){
        for(int j = sx;j <= ex;j++){
            // 현재 종이가 첫 번째 종이와 다름
            if(a[i][j] != num) flag = 0;
        }
    }
}
```

종이의 개수 / 1780

1. 만약 종이가 모두 같은 수로 되어 있다면 이 종이를 그대로 사용한다.
2. (1)이 아닌 경우에는 종이를 같은 크기의 종이 9개로 자르고, 각각의 잘린 종이에 대해서 (1)의 과정을 반복한다.

분할 – 종이가 모두 같은 수로 되어 있지 않으면 9개로 분할

```
if(!flag){
    ll sz = ex - sx + 1;
    for(int i = 0;i < 3;i++){
        for(int j = 0;j < 3;j++){
            ll nsx = sx + sz * i;
            ll nsy = sy + sz * i;
            ll nex = ex + sz * (i + 1) - 1;
            ll ney = ey + sz * (i + 1) - 1;
            dnc(nsx, nsy, nex, ney);
        }
    }
}
```

종이의 개수 / 1780

1. 만약 종이가 모두 같은 수로 되어 있다면 이 종이를 그대로 사용한다.
2. (1)이 아닌 경우에는 종이를 같은 크기의 종이 9개로 자르고, 각각의 잘린 종이에 대해서 (1)의 과정을 반복한다.

정복 – 종이가 모두 같은 수로 되어 있으면 종이의 개수 증가
Base Case -> 종이가 1개

```
if(sx == ex){
    if(a[sx][ex] == -1) return {1, 0, 0};
    else if(a[sx][ex] == 0) return {0, 1, 0};
    return {0, 0, 1};
}
```

종이의 개수 / 1780

1. 만약 종이가 모두 같은 수로 되어 있다면 이 종이를 그대로 사용한다.
2. (1)이 아닌 경우에는 종이를 같은 크기의 종이 9개로 자르고, 각각의 잘린 종이에 대해서 (1)의 과정을 반복한다.

정복 – 종이가 모두 같은 수로 되어 있으면 종이의 개수 증가
문제를 끝까지 분할 했으므로 다시 합쳐야 함

```
tuple <ll, ll, ll> ret = {0, 0, 0};
if(!flag){
    ll sz = ex - sx + 1;
    for(int i = 0;i < 3;i++){
        for(int j = 0;j < 3;j++){
            ll nsx = sx + sz * i;
            ll nsy = sy + sz * i;
            ll nex = ex + sz * (i + 1) - 1;
            ll ney = ey + sz * (i + 1) - 1;
            ret += dnc(nsx, nsy, nex, ney);
        }
    }
}

return ret;
```

종이의 개수 / 1780

전체 코드

```
// {sx, sy}, {ex, ey}의 종이를 볼 때 -1, 0, 1로 채워진 종이의 개수
tuple <ll, ll, ll> dnc(ll sx, ll sy, ll ex, ll ey){
    if(sx == ex){
        if(a[sx][ex] == -1) return {1, 0, 0};
        else if(a[sx][ex] == 0) return {0, 1, 0};
        return {0, 0, 1};
    }

    bool flag = 1; ll num = a[sy][sx];
    for(int i = sy;i <= ey;i++){
        for(int j = sx;j <= ex;j++){
            // 현재 종이가 첫 번째 종이와 다름
            if(a[i][j] != num) flag = 0;
        }
    }

    tuple <ll, ll, ll> ret = {0, 0, 0};
    if(!flag){
        ll sz = ex - sx + 1;
        for(int i = 0;i < 3;i++){
            for(int j = 0;j < 3;j++){
                ll nsx = sx + sz * i;
                ll nsy = sy + sz * i;
                ll nex = ex + sz * (i + 1) - 1;
                ll ney = ey + sz * (i + 1) - 1;
                ret += dnc(nsx, nsy, nex, ney);
            }
        }
    }

    return ret;
}
```

종이의 개수 / 1780

C++

```cpp
#include <iostream>
using namespace std;
using ll = long long;

const ll MAX = 3010;
const ll INF = 1e12;
ll n, a[MAX][MAX], cnt[3];
```

```cpp
int main(){
    ios::sync_with_stdio(0); // fastio
    cin.tie(0), cout.tie(0); // fastio

    cin >> n;
    for(int i = 0;i < n;i++){
        for(int j = 0;j < n;j++) cin >> a[i][j];
    }

    dnc(0, 0, n);
    for(int i = 0;i < 3;i++) cout << cnt[i] << "\n";
    return 0;
}
```

```cpp
void dnc(ll cy, ll cx, ll sz){
    bool flag = 1; // 모든 종이가 같으면 1 아니면 0
    ll num = a[cy][cx]; // 시작 종이의 숫자
    for(int i = cy;i < cy + sz;i++){
        for(int j = cx;j < cx + sz;j++){
            // 시작 종이와 현재 종이가 다르면 flag = 0
            if(a[i][j] != num) flag = 0;
        }
    }

    // 모든 종이가 같으면 시작 종이의 개수를 1 증가
    // 및 재귀 함수 종료
    if(flag){
        cnt[num + 1]++; return;
    }

    // 다음 종이의 크기
    ll nsz = sz / 3;
    for(int i = 0;i < 3;i++){
        for(int j = 0;j < 3;j++){
            // 분할
            dnc(cy + i * nsz, cx + j * nsz, nsz);
        }
    }
}
```

종이의 개수 / 1780

Python

```python
import sys
sys.setrecursionlimit(1000000)
input = sys.stdin.readline

n = int(input())
a = [list(map(int, input().split())) for _ in range(n)]
cnt = [0, 0, 0]
```

```python
def dnc(cy, cx, sz):
    flag = True # 모든 종이가 같으면 1 아니면 0
    num = a[cy][cx] # 시작 종이의 숫자

    for i in range(cy, cy + sz):
        for j in range(cx, cx + sz):
            # 시작 종이와 현재 종이가 다르면 flag = 0
            if a[i][j] != num:
                flag = False

    # 모든 종이가 같으면 시작 종이의 개수를 1 증가
    # 및 재귀 함수 종료
    if flag:
        cnt[num + 1] += 1
        return

    # 다음 종이의 크기
    nsz = sz // 3
    for i in range(3):
        for j in range(3):
            # 분할
            dnc(cy + i * nsz, cx + j * nsz, nsz)

dnc(0, 0, n)
print(cnt[0])
print(cnt[1])
print(cnt[2])
```

종이의 개수 / 1780

시간 복잡도

T(n) = 9 * T(n / 3) + O(n^2)
a = 9, b = 3, d = 2

$$T(n) = \begin{cases} O(n^d), & \text{if } d > \log_b a \\ O(n^d \log n), & \text{if } d = \log_b a \\ O(n^{\log_b a}), & \text{if } d < \log_b a \end{cases}$$

O(n^2 log n)

```
void dnc(ll cy, ll cx, ll sz){
    bool flag = 1; // 모든 종이가 같으면 1 아니면 0
    ll num = a[cy][cx]; // 시작 종이의 숫자
    for(int i = cy;i < cy + sz;i++){
        for(int j = cx;j < cx + sz;j++){
            // 시작 종이와 현재 종이가 다르면 flag = 0
            if(a[i][j] != num) flag = 0;
        }
    }

    // 모든 종이가 같으면 시작 종이의 개수를 1 증가
    // 및 재귀 함수 종료
    if(flag){
        cnt[num + 1]++; return;
    }

    // 다음 종이의 크기
    ll nsz = sz / 3;
    for(int i = 0;i < 3;i++){
        for(int j = 0;j < 3;j++){
            // 분할
            dnc(cy + i * nsz, cx + j * nsz, nsz);
        }
    }
}
```

질문?

Moo 게임 / 5904

백준 5904 / https://www.acmicpc.net/problem/5940

Moo는 술자리에서 즐겁게 할 수 있는 게임이다. 이 게임은 Moo수열을 각 사람이 하나씩 순서대로 외치면 되는 게임이다.

Moo 수열은 길이가 무한대이며, 다음과 같이 생겼다.

```
m o o m o o o m o o m o o o o m o o m o o o m o o m o o o o o
```

Moo 수열은 다음과 같은 방법으로 재귀적으로 만들 수 있다. 먼저, S(0)을 길이가 3인 수열 "m o o"이라고 하자. 1보다 크거나 같은 모든 k에 대해서, S(k)는 S(k-1)과 o가 k+2개인 수열 "m o ... o" 와 S(k-1)을 합쳐서 만들 수 있다.

```
S(0) = "m o o"
S(1) = "m o o m o o o m o o"
S(2) = "m o o m o o o m o o m o o o o m o o m o o o m o o"
```

위와 같은 식으로 만들면, 길이가 무한대인 문자열을 만들 수 있으며, 그 수열을 Moo 수열이라고 한다.

N이 주어졌을 때, Moo 수열의 N번째 글자를 구하는 프로그램을 작성하시오.

입력

첫째 줄에 N ($1 \le N \le 10^9$)이 주어진다.

출력

N번째 글자를 출력한다.

Moo 게임 / 5904

s(0) = “moo”
s(1) = “moo mooo moo”
s(2) = “moo mooo moooo mooo moo”

s(0) = “moo”
s(1) = s(0) + “mooo” + s(0)
s(2) = s(1) + “moooo” + s(1)

예제 입력 1 복사

```
11
```

예제 출력 1 복사

```
m
```

Moo 게임 / 5904

s(0) = “moo”
s(1) = s(0) + “mooo” + s(0)
s(2) = s(1) + “moooo” + s(1)

분할 – 크기를 하나 줄이고 인덱스 조절

앞의 구간이면 인덱스는 그대로
뒤의 구간이면 앞의 구간의 크기를 빼줘야 함

Moo 게임 / 5904

s(0) = “moo”
s(1) = s(0) + “mooo” + s(0)
s(2) = s(1) + “moooo” + s(1)

뒤의 구간이면 앞의 구간의 크기 + “moo..”의 크기를 빼줘야 함

-> 구간의 크기를 저장
len[0] = 3, len[i] = len[i – 1] + i + 3

Moo 게임 / 5904

s(0) = “moo”
s(1) = s(0) + “mooo” + s(0)
s(2) = s(1) + “moooo” + s(1)

정복 - 가운데 구간일 때 첫번째면 ‘m’ 아니면 ‘o’ 반환

Moo 게임 / 5904

s(0) = “moo”
s(1) = s(0) + “mooo” + s(0)
s(2) = s(1) + “moooo” + s(1)

Base Case
인덱스가 1이면 ‘m’ 아니면 ‘o’

Moo 게임 / 5904

C++

```cpp
#include <iostream>
using namespace std;
using ll = long long;

const ll MAX = 55;
const ll INF = 1e12;
ll n, len[MAX], mx;
```

```cpp
int main(){
    ios::sync_with_stdio(0); // fastio
    cin.tie(0), cout.tie(0); // fastio

    cin >> n; len[0] = 3;
    for(int i = 1;i < 55;i++){
        len[i] = 2 * len[i - 1] + i + 3; mx = i;
        // 오버플로우 방지
        if(len[i] > n) break;
    }

    cout << dnc(n, mx);
    return 0;
}
```

```cpp
char dnc(ll cur, ll sz){
    // Base Case
    if(!sz) return (cur == 1 ? 'm' : 'o');

    // 앞쪽 구간이면 인덱스는 그대로
    if(cur <= len[sz - 1]) return dnc(cur, sz - 1);

    // 뒤쪽 구간이면 이전의 크기 + 'moo..'의 크기를 더한 만큼 인덱스를 뺌
    else if(cur > len[sz - 1] + sz + 3){
        return dnc(cur - len[sz - 1] - sz - 3, sz - 1);
    }

    // 가운데 구간이고 첫번째 인덱스면 'm' 반환 아니면 'o' 반환
    return (cur == len[sz - 1] + 1 ? 'm' : 'o');
}
```

Moo 게임 / 5904

Python

```python
import sys
sys.setrecursionlimit(1000000)
input = sys.stdin.readline

MAX = 55
len = [0] * MAX
mx = 0
```

```python
n = int(input().strip())
len[0] = 3
for i in range(1, 55):
    len[i] = 2 * len[i - 1] + i + 3
    mx = i
    # 오버플로우 방지
    if len[i] > n:
        break

print(dnc(n, mx))
```

```python
def dnc(cur, sz):
    # Base Case
    if not sz:
        return 'm' if cur == 1 else 'o'

    # 앞쪽 구간이면 인덱스는 그대로
    if cur <= len[sz - 1]:
        return dnc(cur, sz - 1)

    # 뒤쪽 구간이면 이전의 크기 + "moo.."의 크기를 더한 만큼 인덱스를 뺌
    elif cur > len[sz - 1] + sz + 3:
        return dnc(cur - len[sz - 1] - sz - 3, sz - 1)

    # 가운데 구간이고 첫번째 인덱스면 'm' 아니면 'o' 반환
    return 'm' if cur == len[sz - 1] + 1 else 'o'
```

Moo 게임 / 5904

시간 복잡도

$T(m) = T(m - 1) + O(1)$
$\Rightarrow O(m)$

```
char dnc(ll cur, ll sz){
    // Base Case
    if(!sz) return (cur == 1 ? 'm' : 'o');

    // 앞쪽 구간이면 인덱스는 그대로
    if(cur <= len[sz - 1]) return dnc(cur, sz - 1);

    // 뒤쪽 구간이면 이전의 크기 + 'moo..'의 크기를 더한 만큼 인덱스를 뺌
    else if(cur > len[sz - 1] + sz + 3){
        return dnc(cur - len[sz - 1] - sz - 3, sz - 1);
    }

    // 가운데 구간이고 첫번째 인덱스면 'm' 반환 아니면 'o' 반환
    return (cur == len[sz - 1] + 1 ? 'm' : 'o');
}
```

질문?

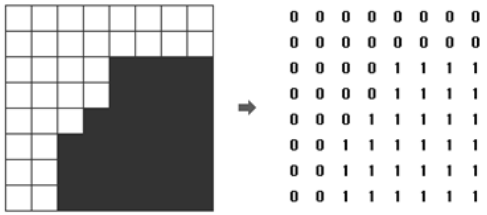
쿼드트리 / 1992

백준 1992 / https://www.acmicpc.net/problem/1992

문제

흑백 영상을 압축하여 표현하는 데이터 구조로 쿼드 트리(Quad Tree)라는 방법이 있다. 흰 점을 나타내는 0과 검은 점을 나타내는 1로만 이루어진 영상(2차원 배열)에서 같은 숫자의 점들이 한 곳에 많이 몰려있으면, 쿼드 트리에서는 이를 압축하여 간단히 표현할 수 있다.

주어진 영상이 모두 0으로만 되어 있으면 압축 결과는 "0"이 되고, 모두 1로만 되어 있으면 압축 결과는 "1"이 된다. 만약 0과 1이 섞여 있으면 전체를 한 번에 나타내지를 못하고, 왼쪽 위, 오른쪽 위, 왼쪽 아래, 오른쪽 아래, 이렇게 4개의 영상으로 나누어 압축하게 되며, 이 4개의 영역을 압축한 결과를 차례대로 괄호 안에 묶어서 표현한다



위 그림에서 왼쪽의 영상은 오른쪽의 배열과 같이 숫자로 주어지며, 이 영상을 쿼드 트리 구조를 이용하여 압축하면 "`(0(0011)(0(0111)01)1)`"로 표현된다. N ×N 크기의 영상이 주어질 때, 이 영상을 압축한 결과를 출력하는 프로그램을 작성하시오.

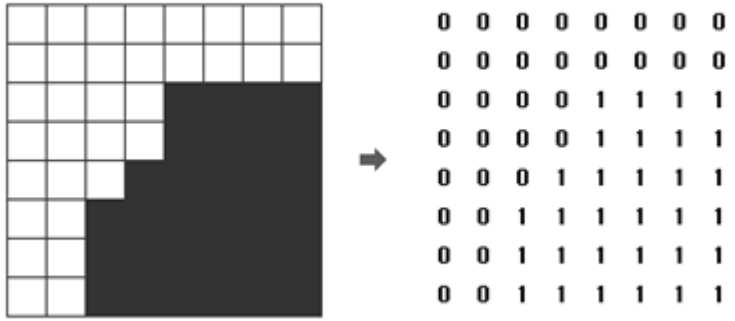
입력

첫째 줄에는 영상의 크기를 나타내는 숫자 N 이 주어진다. N 은 언제나 2의 제곱수로 주어지며, 1 ≤ N ≤ 64의 범위를 가진다. 두 번째 줄부터는 길이 N의 문자열이 N개 들어온다. 각 문자열은 0 또는 1의 숫자로 이루어져 있으며, 영상의 각 점들을 나타낸다.

출력

영상을 압축한 결과를 출력한다.

쿼드트리 / 1992



```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 1
0 0 0 0 1 1 1 1
0 0 0 1 1 1 1 1
0 0 1 1 1 1 1 1
0 0 1 1 1 1 1 1
0 0 1 1 1 1 1 1
```

"(0(0011)(0(0111)01)1)"

예제 입력 1 복사

```
8
11110000
11110000
00011100
00011100
11110000
11110000
11110011
11110011
```

예제 출력 1 복사

```
((110(0101))(0010)1(0001))
```

쿼드트리 / 1992

C++

```cpp
#include <iostream>
using namespace std;
using ll = long long;

const ll MAX = 66;
const ll INF = 1e12;
ll n;
string st[MAX];
```

```cpp
int main(){
    ios::sync_with_stdio(0); // fastio
    cin.tie(0), cout.tie(0); // fastio

    cin >> n;
    for(int i = 0;i < n;i++) cin >> st[i];
    dnc(0, 0, n);
}
```

```cpp
void dnc(ll cy, ll cx, ll sz){
    bool flag = 1; // 모든 숫자가 같으면 1 아니면 0
    char num = st[cy][cx]; // 시작 숫자
    for(int i = cy;i < cy + sz;i++){
        for(int j = cx;j < cx + sz;j++){
            // 시작 숫자와 현재 숫자가 다르면 flag = 0
            if(st[i][j] != num) flag = 0;
        }
    }

    if(flag){
        cout << num;
        return;
    }

    cout << "(";

    // 분할
    ll nsz = sz / 2;
    for(int i = 0;i < 2;i++){
        for(int j = 0;j < 2;j++){
            dnc(cy + i * nsz, cx + j * nsz, nsz);
        }
    }

    cout << ")";
}
```

쿼드트리 / 1992

Python

```python
import sys
sys.setrecursionlimit(1000000)
input = sys.stdin.readline

MAX = 66
INF = 10**12
n = 0
st = []
```

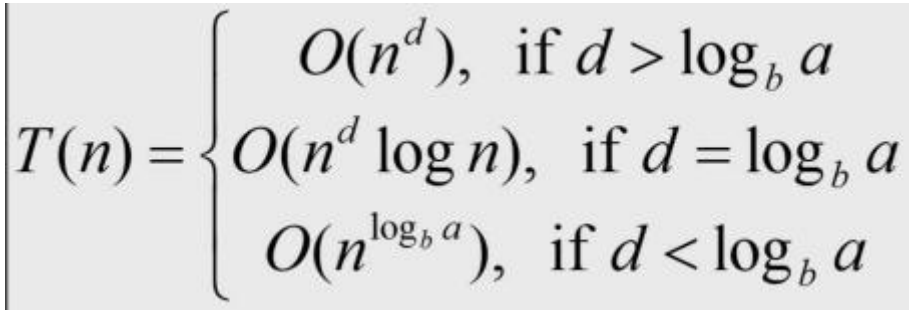
```python
n = int(input().strip())
st = [input().strip() for _ in range(n)]
dnc(0, 0, n)
```

```python
def dnc(cy, cx, sz):
    flag = True # 모든 숫자가 같으면 1 아니면 0
    num = st[cy][cx] # 시작 숫자
    for i in range(cy, cy + sz):
        for j in range(cx, cx + sz):
            # 시작 숫자와 현재 숫자가 다르면 flag = 0
            if st[i][j] != num:
                flag = False
    if flag:
        print(num, end = '')
        return

    print("(", end = '')
    nsz = sz // 2

    # 분할
    for i in range(2):
        for j in range(2):
            dnc(cy + i * nsz, cx + j * nsz, nsz)
    print(")", end = '')
```

쿼드트리 / 1982

시간 복잡도

T(n) = 4 * T(n / 2) + O(n^2)
a = 4, b = 2, d = 2



$$T(n) = \begin{cases} O(n^d), & \text{if } d > \log_b a \\ O(n^d \log n), & \text{if } d = \log_b a \\ O(n^{\log_b a}), & \text{if } d < \log_b a \end{cases}$$

O(n^2 log n)

```
void dnc(ll cy, ll cx, ll sz){
    bool flag = 1; // 모든 숫자가 같으면 1 아니면 0
    char num = st[cy][cx]; // 시작 숫자
    for(int i = cy;i < cy + sz;i++){
        for(int j = cx;j < cx + sz;j++){
            // 시작 숫자와 현재 숫자가 다르면 flag = 0
            if(st[i][j] != num) flag = 0;
        }
    }

    if(flag){
        cout << num;
        return;
    }

    cout << "(";

    // 분할
    ll nsz = sz / 2;
    for(int i = 0;i < 2;i++){
        for(int j = 0;j < 2;j++){
            dnc(cy + i * nsz, cx + j * nsz, nsz);
        }
    }

    cout << ")";
}
```

질문?

기본 과제

종이의 개수 – https://www.acmicpc.net/problem/1780

Moo 게임 - https://www.acmicpc.net/problem/5904

쿼드트리 – https://www.acmicpc.net/problem/1992

Z – https://www.acmicpc.net/problem/1074

별 찍기 10 - https://www.acmicpc.net/problem/2447

고생하셨습니다