dominator.cpp

```cpp
#include<bits/stdc++.h>

using namespace std;

vector< vector < pair <int, int> > > graph;
vector<int> reachables, domination;
int N, c;

void initial_bfs(int root) {
    reachables[root] = 1;
    queue<int> q;
    int u;
    q.push( root );
    while ( !q.empty() ){
        u = q.front();
        q.pop();
        for (int i = 0; i < graph[u].size(); i++){
            int neighbor = graph[u][i].first;
            if ( reachables[ neighbor ] == 0){
                q.push( neighbor );
                reachables[neighbor] = 1;
            }
        }
    }
}

void bfs(int root, int ignore) {
    if (root == ignore) return;
    domination[root] = 1;
    queue<int> q;
    int u;
    q.push( root );
    while ( !q.empty() ){
        u = q.front();
        q.pop();
        for (int i = 0; i < graph[u].size(); i++){
            int neighbor = graph[u][i].first;
            if ( domination[ neighbor ] == 0 && neighbor != ignore){
                q.push( neighbor );
                domination[neighbor] = 1;
            }
        }
    }
}

void pline() {
    printf("+");
    for (int i = 1; i < N * 2; i++)
        printf("-");
    printf("+\n");
}

int main(){
    cin.tie(0);
    ios_base::sync_with_stdio(0);

    //freopen("input.txt", "r", stdin);

    int T, tmp;
    cin >> T;
    while ( c < T ){
```

```
        c++;
        cin >> N;
        graph.assign(N, vector< pair<int, int > >());
        for (int i = 0; i < N; i++){
            for ( int j = 0; j < N; j++){
                cin >> tmp;
                if (tmp) graph[i].push_back( pair<int, int>( j, 1 ) );
            }
        }
        reachables.assign(N, 0);
        initial_bfs(0);
        printf("Case %d:\n", c);
        for (int i = 0; i < N; i++){
            domination.assign(N, 0);
            pline();
            printf("|");
            bfs(0, i);
            for (int j = 0; j < N; j++){
                if ( domination[j] == 0 && reachables[j] == 1)
                    printf("Y|");
                else printf("N|");
            }
            printf("\n");
        }
        pline();
    }
}
```