```
transpose([], []).
transpose([F|Fs], Ts) :-
    transpose(F, [F|Fs], Ts).

transpose([], _, []).
transpose([_|Rs], Ms, [Ts|Tss]) :-
    lists_firsts_rests(Ms, Ts, Ms1),
    transpose(Rs, Ms1, Tss).

lists_firsts_rests([], [], []).
lists_firsts_rests([[F|Os]|Rest], [F|Fs], [Os|Oss]) :-
    lists_firsts_rests(Rest, Fs, Oss).

append([L], L).
append([H|T], L) :-
    append(T, LSub),
    append(H, LSub, L).

same_length([], []).
same_length([_|T1], [_|T2]) :- same_length(T1, T2).

rev([], []).
rev([H1|T1], [H2|T2]) :-
    reverse(H1, H2),
    rev(T1, T2).

max_val(0, []).
max_val(E, [H|T]) :-
    max_val(SubE, T),
    E = max(SubE, H).

constrain_right(_, [_], Count) :-
    fd_domain(Count, 1, 1).
constrain_right(N, [H|T], Count) :-
    max_val(M, T),
    H #># M,
    constrain_right(N, T, SubCount),
    fd_max(SubCount, Val),
    Incr is Val+1,
    fd_domain(Count, Incr, Incr).

constrain_right(N, [H|T], Count) :-
    max_val(M, T),
    H #<# M,
    constrain_right(N, T, Count).

visible_right(N, R, Count) :-
    length(R, N),
    fd_domain(R, 1, N),
    fd_domain(Count, 1, N),
    fd_all_different(R),
    constrain_right(N, R, Count),
    fd_labeling(R),
    fd_labeling(Count).

counts_right(_, [], []).
counts_right(N, [H|T], [C|R]) :-
    visible_right(N, H, C),
    counts_right(N, T, R).

tower(0, [], counts([], [], [], [])).
tower(N, T, counts(U, D, L, R)) :-
    length(T, N),
    maplist(same_length(T), T),
    append(T, Vs),
    fd_domain(Vs, 1, N),
    transpose(T, Trans),
    rev(T, TR),
    rev(Trans, TransR),
    maplist(fd_all_different, T),
    maplist(fd_all_different, Trans),
    counts_right(N, T, R),
    counts_right(N, TR, L),
    counts_right(N, Trans, D),
    counts_right(N, TransR, U),
    fd_labeling(Vs).

plain_constrain_right(_, [_], 1).
plain_constrain_right(N, [H|T], Count) :-
    max_list(T, M),
    H > M,
    plain_constrain_right(N, T, SubCount),
    Count is SubCount+1.

plain_constrain_right(N, [H|T], Count) :-
    max_list(T, M),
    H < M,
    plain_constrain_right(N, T, Count).

generate_domain(L, Min, Max) :-
    findall(Num, between(Min, Max, Num), L).

all_different([]).
all_different([_]).
all_different([F, S|T]) :-
    \+(F=S),
    all_different([F|T]),
    all_different([S|T]).

generate_unique_list_domain([], _, _).
generate_unique_list_domain([H|T], Min, Max) :-
    generate_unique_list_domain(T, Min, Max),
    generate_domain(Dom, Min, Max),
    member(H, Dom),
    all_different([H|T]).

plain_visible_left_right(N, R, LCount, RCount) :-
    length(R, N),
    generate_unique_list_domain(R, 1, N),
    generate_domain(Dom, 1, N),
    member(RCount, Dom),
    plain_constrain_right(N, R, RCount),
    member(LCount, Dom),
    reverse(R, Rev),
    plain_constrain_right(N, Rev, LCount).

plain_counts_left_right(_, _, [], [], []).
plain_counts_left_right(N, Rows, [H|T], [LC|LR], [RC|RR]) :-
    SubRows is Rows-1,
    plain_visible_left_right(N, H, LC, RC),
```

```
    plain_counts_left_right(N, SubRows, T, LR, RR).

plain_counts_left_right(N, B, LCounts, RCounts) :-
    length(B, N),
    length(LCounts, N),
    length(RCounts, N),
    plain_counts_left_right(N, N, B, LCounts, RCounts).

plain_tower(N, T, counts(U, D, L, R)) :-
    plain_counts_left_right(N, T, L, R),
    transpose(T, Trans),
    plain_counts_left_right(N, Trans, U, D).

speedup(S) :-
    statistics(_, _),
    tower(4, _,
          counts([_, 2, _, _],
                 [3, _, _, _],
                 [_, 2, _, 3],
                 [_, _, 3, _])),
    statistics(user_time, [_, TTime]),
    plain_tower(4, _,
                counts([_, 2, _, _],
                       [3, _, _, _],
                       [_, 2, _, 3],
                       [_, _, 3, _])),
    statistics(user_time, [_, PTTime]),
    S is PTTime/TTime, !.

ambiguous(N, C, T1, T2) :-
    tower(N, T1, C),
    tower(N, T2, C),
    \+(T1=T2), !.
```