```
#include <bits/stdc++.h>
#define FR(i, n) for (int i = 0; i < (n); ++i)
using namespace std;

typedef long long ll;
typedef vector<int> vi;
typedef pair<int, int> ii;

typedef vector<ii> vii;
typedef vector<vi> vvi;

// Binomial Coef, non-dp method
// note:
// C(n, k) == C(n, n - k);
// C(n, 0) = C(n, n) = 1;
// C(n, k) = C(n - 1, k - 1) + C(n - 1, k);
// recursively, in DP method, as
// for(i=0;i<=MAXN;i++)for(j=0;j<=i;j++)dp[i][j]=C(i,j);
// many probs, computing all not necessary or impractical to
// store, so use product of sequence form
ll C(int n, int k) {
   if (k == 0 || k == n)
      return 1;

   k = min(k, n - k);
   ll ans = 1;
   for (ll i = 1; i <= k; i++)
      ans = ans * (n - k + i) / i;
   return ans;
}

const int MM = 20; // max len, lcs
int lcsAndPath(int A[MM], int a, int B[MM], int b, int ans[MM]) {
   int L[MM + 1][MM + 1];
   for (int i = a; i >= 0; i--)
      for (int j = b; j >= 0; j--)
     if (i == a || j == b)
        L[i][j] = 0;
     else if (A[i] == B[i])
        L[i][j] = 1 + L[i + 1][j + 1];
     else
        L[i][j] = max(L[i + 1][j], L[i][j + 1]);

   int i = 0, j = 0, k = 0;
   while (i < a && j < b) {
      if (A[i] == B[j])
     ans[k++] = A[i], i++, j++;
      else if (L[i + 1][j] > L[i][j + 1])
     i++;
      else if (L[i + 1][j] < L[i][j + 1])
     j++;
      else
     j++; // tiebreaker
   }
   return L[0][0]; // len, ans has actual values as the path
}

// All topsort
void printAllTS(vi &res, vii &vind, vvi &g) {
   bool d = true;
   for (int i = 0; i < g.size(); i++) {
      if (vind[i] == ii(0, 0)) {
    for (auto &j : g[i]) vind[j].second--;
    vind[i] = ii(1, 0);
    res.push_back(i);
    printAllTS(res, vind, g);
    for (auto &j : g[i]) vind[j].second++;
    vind[i] = ii(0, 0);
    res.pop_back();
    d = false;
      }
   }
   if (d) {
      for (auto &i : res) cout << i << " ";
      cout << endl;
   }
}

void allTS(vvi &g) {
   vii vind(g.size(), ii(0, 0)); // 2nd is pair (visible, indegree)
   for (int i = 0; i < g.size(); i++)
      for (auto &v : g[i])
    vind[v].second++;
   vi res;
   printAllTS(res, vind, g);
}

// compute fibonacci, O(logn)
// fib(46) last to fit in int (32-bit signed), else change to ll
int FIB[1000] = { 0 };
int fib(int n) {
   if (n < 2) return (FIB[n] = n);
   if (FIB[n]) return FIB[n];

   int k = (n & 1) ? (n + 1) / 2 : n / 2;
   FIB[n] = (n & 1) ? fib(k) * fib(k) + fib(k - 1) * fib(k - 1)
      : (2 * fib(k - 1) + fib(k)) * fib(k);
   return FIB[n];
}

int modexp(int x, int n, int m) {
   if (n == 0)
      return 1;
   if (n & 1)
      return ((x % m) * modexp(x, n - 1, m)) % m;

   int y = modexp(x, n / 2, m);
   return (y * y) % m;
}

int main() {

   ios_base::sync_with_stdio(false);
   cin.tie(NULL);

   int n;
   cin >> n;
```

```
    vvi g(n, vi());

    int u, v;
    while (cin >> u >> v) {
        g[u].push_back(v);
    }

    allTS(g);

}
```