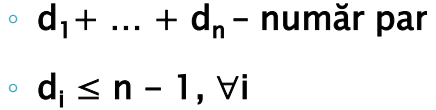
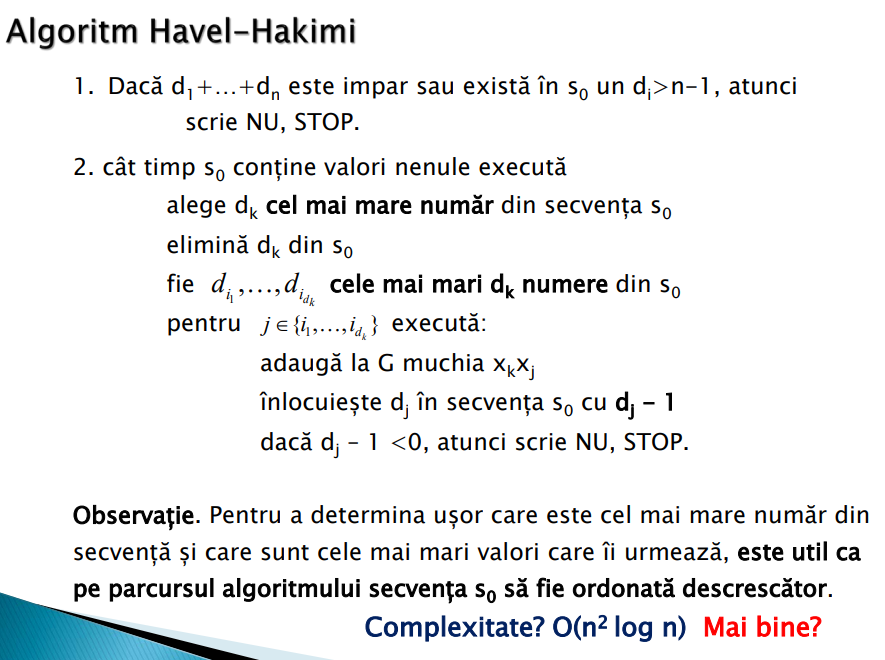
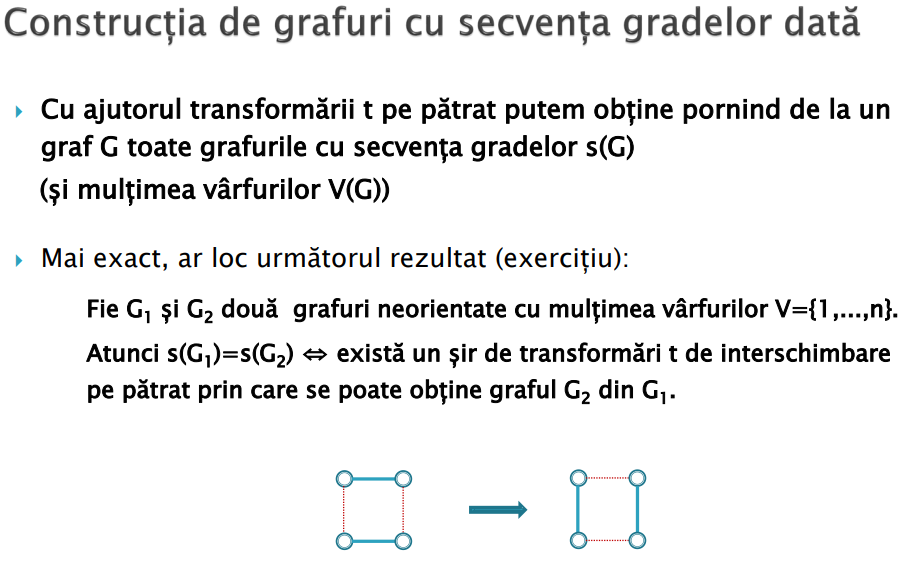


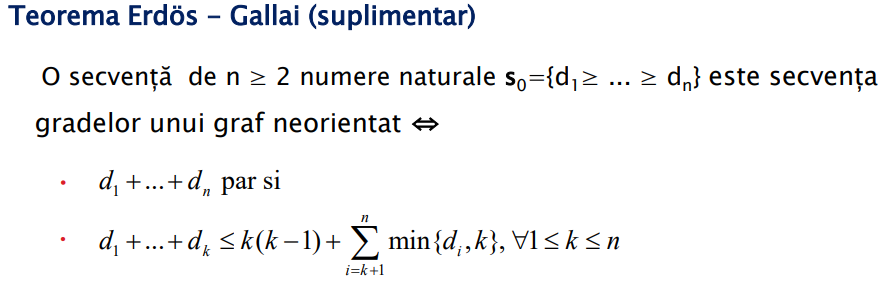
**Conditii necesare** sa se poata forma un graf G din secventa de grade data:



**Conditii suficiente** :

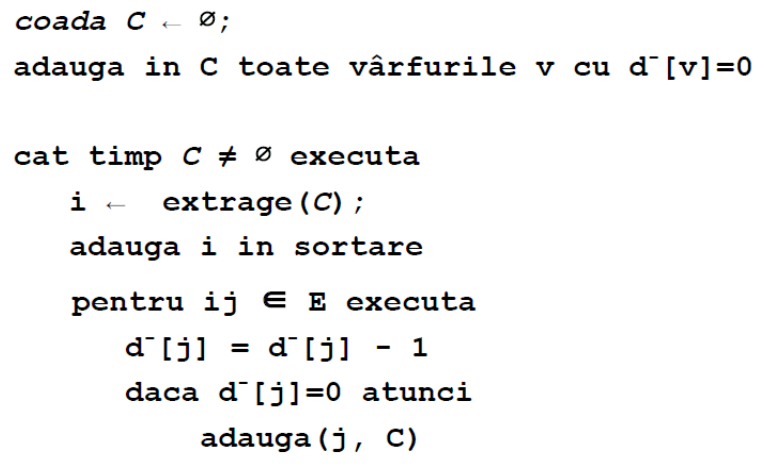






**Sortare topologica**

* fol in pb de ordine, det drumuri ciclice
* A topological ordering is possible if and only if the graph **has no directed cycles**, that is, if it is a directed acyclic graph (DAG). **Any DAG has at least one topological ordering**.

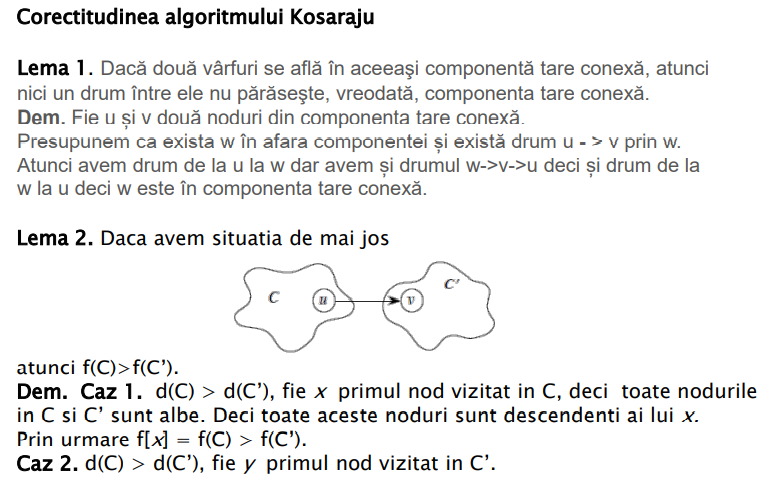


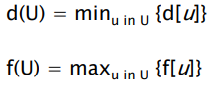
**Componente tare conexe** -> O(N+M)

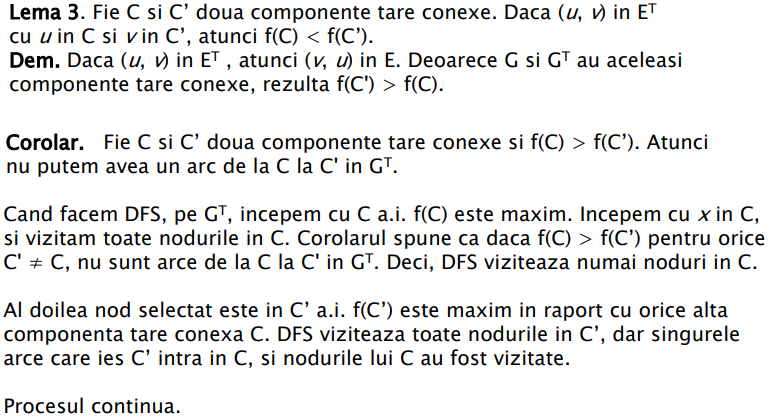
in graf orientat

* Folosind doar două parcurgeri, una în G și una în GT (deci slower) ⇒ Algoritmul lui Kosaraju
* Folosind o singură parcurgere ( idee similară cu cea de la comp biconexe) ⇒ Algoritmul lui Tarjan

<https://www.geeksforgeeks.org/comparision-between-tarjans-and-kosarajus-algorithm/>



unde : => f = timp de finalizare maxim, U = CTC



int n;

vector<int> la[Nmax];

stack<int> st[Nmax];

int disc[Nmax], low[Nmax];

void Tarjan (int u) {

int time = 0;

disc[u] = low[u] = ++time;

st.push(u);

viz[u] = 1;

for (int i = 0; i < la.size(); i++) {

int v = i; // v is current adjacent of 'u'

      if (disc[v] == -1) // If v is not visited yet, then recur for it

      {

            Tarjan(v);

            // Check if the subtree rooted with 'v' has a connection to one of the

ancestors of 'u'

            low[u] = min(low[u], low[v]); // Case 1

      }

      // Update low value of 'u' only of 'v' is still in stack (i.e. it's a back edge, not cross edge).

      else if (viz[v] == true) // Case 2

            low[u] = min(low[u], disc[v]);

}

     // head node found, pop the stack and print an SCC

     int w = 0; // To store stack extracted vertices

     if (low[u] == disc[u])

     {

         while (st->top() != u)

         {

         w = (int) st->top();

             cout << w << " ";

             viz[w] = 0;

             st->pop();

         }

         w = (int) st->top();

         cout << w << "\n";

         viz[w] = 0;

         st->pop();

    }

}

}

int main() {

 for (int i = 0; i < la.size(); i++)

        if (disc[i] == 0)

            Tarjan (i);

}