Design and Analysis of Algorithms

Mohammad GANJTABESH

mgtabesh@ut.ac.ir

School of Mathematics, Statistics and Computer Science,
University of Tehran,
Tehran, Iran.

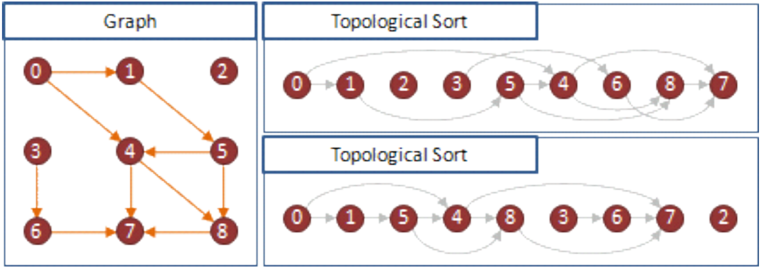
Graph Theoretical Problems

- Basec Definitions
- Graph Representation
- Graph Traversal (BFS, DFS)
- Topological Sort
- Strongly Connected Components
- Shortest Paths
  - Single-Source All Destination (Bellman-Ford, Dijkestra)
  - All-Pairs (Floyd-Warshall, Johnson)
- Minimum Spanning Tree (Kruskal, Prim)

Topological Sort

Topological Sort

A **topological sort** of a directed acyclic graph $G = (V, E)$ is a linear ordering of all its vertices such that if G contains an edge (u, v), then u appears before v in the ordering.



Topological Sort

- **First Approach:** Repeatedly find a vertex with 0 indegree and remove it with all its out going edges.
- **Second Approach:** Use DFS as follows.

TOPOLOGICAL-SORT(G)

1 call DFS(G) to compute finishing times $f[v]$ for each vertex v
2 as each vertex is finished, insert it onto the front of a linked list
3 **return** the linked list of vertices

Lemma

A directed graph G is acyclic if and only if a depth-first search of G yields no back edges.

Proof.

$\Rightarrow$: Suppose that (u, v) is back edge $\Longrightarrow$ v is an ancestor of u $\Longrightarrow$ there is a path from v to u in G $\Longrightarrow$ there is a cycle in G (contradiction).
$\Leftarrow$: Suppose that c is a cycle in G, v be the first vertex to be discovered in c, and u, v be the preceding edge in c. At time $d[v]$, the vertices of c form a path of white vertices from v to u. So, vertex u becomes a descendant of v in the depth-first forest. Therefore, (u, v) is a back edge. $\square$

Topological Sort

Theorem

TOPOLOGICAL-SORT(G) *produces a topological sort of a directed acyclic graph G.*

Proof.

It suffices to show that $(u, v) \in E$ then $f[v] < f[u]$. When (u, v) is explored, v cannot be gray (if so, (u, v) is back edge and contradicting previous lemma). Therefore, v must be either white or black.

- If v is white, it becomes a descendant of u, and so $f[v] < f[u]$.
- If v is black, it has already been finished, so that $f[v]$ has already been set and $f[v] < f[u]$.

$\square$

Strongly connected components

Recall

A strongly connected component of a directed graph $G = (V, E)$ is a maximal set of vertices $C \subset V$ such that for every pair of vertices u and v in C, we have both $u \rightsquigarrow v$ and $v \rightsquigarrow u$.

- Decomposing a directed graph into its strongly connected components.
- Many algorithms that work with directed graphs begin with such a decomposition.

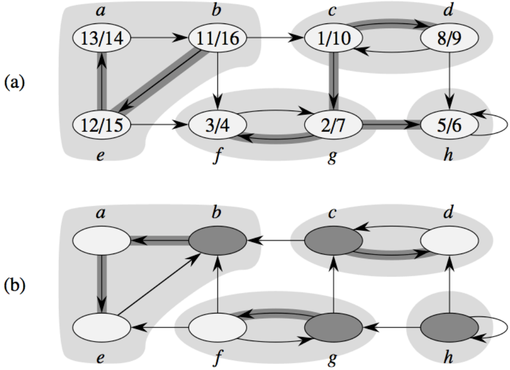
Definition

For a given graph $G = (V, E)$, the transpose of G is a graph $G^T = (V, E^T)$, where $E^T = \{(u, v) \, : \, (v, u) \in E\}$.

The transpose of G can be constructed in $O(|V| + |E|)$.

Strongly connected components

The graphs G and G^T have exactly the same strongly connected components: u and v are reachable from each other in G if and only if they are reachable from each other in G^T.



Strongly connected components

The following $\Theta(V+E)$-time algorithm computes the strongly connected components of a directed graph $G=(V,E)$ using two depth-first searches, one on G and one on G^T.

STRONGLY-CONNECTED-COMPONENTS(G)

1 call DFS(G) to compute finishing times $f[u]$ for each vertex u
2 compute G^{T}
3 call DFS(G^{T}), but in the main loop of DFS, consider the vertices
 in order of decreasing $f[u]$ (as computed in line 1)
4 output the vertices of each tree in the depth-first forest formed in line 3 as a
 separate strongly connected component

Strongly connected components

Lemma

Let C and C' be distinct strongly connected components in directed graph $G = (V, E)$, let $u, v \in C$, let $u', v' \in C'$, and suppose that there is a path $u \leadsto u'$ in G. Then there cannot also be a path $v' \leadsto v$ in G.

Proof.

If there is a path $v' \leadsto v$ in G, then there are paths $u \leadsto u' \leadsto v'$ and $v' \leadsto v \leadsto u$ in G. Thus, u and v' are reachable from each other, thereby contradicting the assumption that C and C' are distinct strongly connected components. □

Definition

For a given graph $G = (V, E)$, suppose that $U \subset V$. Then we define the earliest discovery time of U by $d(U) = min_{u \in U}\{d[u]\}$ and latest finishing time by $f(U) = max_{u \in U}\{f[u]\}$.

Strongly connected components

Lemma

Let C and C' be distinct strongly connected components in directed graph $G = (V, E)$. Suppose that there is an edge $(u, v) \in E$, where $u \in C$ and $v \in C'$. Then $f(C) > f(C')$.

Proof.

There are two cases:

- $d(C) < d(C')$:
  - Let x be the first vertex discovered in C.
  - At time $d[x]$, all vertices in C and C' are white.
  - $(u, v) \in E \Longrightarrow \forall w \in C'$ we have $x \rightsquigarrow u \longrightarrow v \rightsquigarrow w$.
  - All vertices in C and C' become descendants of x.
  - $f[x] = f(C) > f(C')$.
- $d(C) > d(C')$:
  - Let y be the first vertex discovered in C'.
  - At time $d[y]$, all vertices in C' and C are white.
  - All vertices in C' become descendants of $y \Longrightarrow f[y] = f(C')$.
  - There is no path from C' to C.
  - No vertex in C is reachable from y and so at time $f[y]$ all vertices in C are still white.
  - Thus, $\forall w \in C$, we have $f[w] > f[y] \Longrightarrow f(C) > f(C')$. $\square$

Strongly connected components

Theorem

STRONGLY-CONNECTED-COMPONENTS(G) *correctly computes the strongly connected components of a directed graph G.*

Proof.

The proof is based on induction on the number of depth-first trees performed in line 3. The inductive hypothesis is that the first k trees produced in line 3 are strongly connected components. The basis for the induction, when $k = 0$, is trivial.

- Consider the $(k+1)$st produced tree.
- Let the root of this tree be vertex u, and let u be in strongly connected component C.
- $f[u] = f(C) > f(C')$ for any strongly connected component C' other than C that has been visited.
- At time $d[u]$, all other vertices of C are white and they are descendants of u.
- Any edges in G^T that leave C must be to strongly connected components that have already been visited (previous lemma).
- Thus, no vertex in any strongly connected component other than C will be a descendant of u during the depth-first search of G^T.
- Therefore, the vertices of the depth-first tree in G^T that is rooted at u form exactly one strongly connected component.

$\square$

Exercises

1. Give a linear-time algorithm that takes as input a directed acyclic graph $G = (V, E)$ and two vertices s and t, and returns the number of paths from s to t in G.
2. Give an algorithm that determines whether or not a given undirected graph $G = (V, E)$ contains a cycle. Your algorithm should run in $O(V)$ time, independent of $|E|$.
3. How can the number of strongly connected components of a graph change if a new edge is added?
4. Given a directed graph $G = (V, E)$, explain how to create another graph $G^{'} = (V, E^{'})$ such that (a) $G^{'}$ has the same strongly connected components as G, and (b) $E^{'}$ is as small as possible. Describe a fast algorithm to compute $G^{'}$.
5. A directed graph $G = (V, E)$ is said to be semiconnected if, for all pairs of vertices $u, v \in V$, we have $u \rightsquigarrow v$ or $v \rightsquigarrow u$. Give an efficient algorithm to determine whether or not G is semiconnected. Prove that your algorithm is correct and analyze its running time.