Graphs, Search, Components

- Cut vertex
- Block
- Edge
- Connected component
- Vertex
- Isolated vertex
- Bridge
- Bridge component

Undirected Graph

Representations:

Adjacency Matrix

\[
\begin{array}{cccc}
  a & b & c & d \\
  0 & 1 & 0 & 1 \\
  b & 1 & 0 & 1 \\
  c & 0 & 1 & 0 \\
  d & 1 & 0 & 0 \\
\end{array}
\]

Adjacency Lists

- a: b, d
- b: a, c
- c: b
- d: a
Directed Graph

topological order: \( i \to j \Rightarrow n(i) < n(j) \)
Search

Explore vertices systematically by traversing edges

Mark vertices when visited

Traverse an untraversed edge from a visited vertex or else

Start a new search from an unvisited vertex

Breadth-first search: queue of visited vertices

Depth-first search: stack of visited vertices

\[
dfs(v) : \{ \text{previsit}(v); \text{scan}(v); \text{postvisit}(v) \}
\]

\[
\text{scan}(v) : \text{for } e \in \text{arcs out}(v) \rightarrow \text{traverse}(e)
\]

\[
\text{traverse}(e) : \{ \text{advance}(e); \\
\quad \text{if not previsited}(t(e)) \rightarrow \text{dfs}(t(e)); \\
\quad \text{retreat}(e) \}
\]

\[
\text{explore}(G) : \text{for } v \in G \rightarrow \text{if not previsited}(v) \rightarrow \text{dfs}(v)
\]
Strong Components by DFS (Gabow)

Maintain stack of tentative components

Add each new vertex to stack as a singleton component

When advancing along an edge, if it leads from a component to a lower component on stack, combine all components down to the lower component

When postvisiting the last vertex in a component, output the component

Needs disjoint set union to maintain components:

$O(m+n)\alpha(n)$

Components output in reverse topological order
Linear-Time Version

Maintain stack of vertices not in permanent components in preorder

Observe: tentative components are intervals on this stack

Maintain separate stack of (indices of) bottom vertices in tentative components

vertex number: 0 if not previsited

  stack position if positive

  - component number if negative
Blocks

Number vertices in pre(visit) order

$\text{low}(v) = \min \{ \text{pre}(v) \} \cup \{ \text{pre}(w) \mid \exists (x,w) \text{ with } x \text{ a descendant of } v \}$

$v$ a cut vertex iff start vertex with degree $\geq 2$

or $\exists \text{ child } w$ of $v$ with $\text{pre}(v) \leq \text{low}(w)$

Algorithm

Initialize $\text{low}(v) = \text{pre}(v) \forall v$, stack empty

$\text{advance}(v,w)$: add $(v,w)$ to stack

$\text{retreat}(v,w)$: if $(v,w)$ not a tree edge →

$\text{low}(v) = \min \{ \text{low}(v), \text{pre}(w) \}$

else $\text{low}(v) = \min \{ \text{low}(v), \text{low}(w) \}$

if $\text{low}(w) \geq \text{pre}(v)$, pop edges
down to and including $(v,w)$
from stack to form a block