Algorithm Design & Analysis (CSE222)

Lecture-15

Recap

- Strongly Connected
  - Two vertices u and v are strongly connected if there is path from u to v and v to u.
  - If every pairs of vertices are strongly connected then the graph is a strongly connected graph.
  - Strong component (or scc) is the maximal strongly connected subgraph of G.
  - Condensation of a directed graph is always a DAG.
- Fix a DFS traversal, each SC has a vertex whose parent is either in another SC or it has no parents.
  - In C, show that every vertex in a path v to u are in C.
  - v be the first vertex in C, then for every w in C is its descendant & there is a path from v to w.
  - So, parent(w) $\in$ C.
- Number of SC
  - Vertices in condensation graphs are SC.
  - Find sink component, remove it and increment SC count and repeat.
  - #SC(G) = #SC(G^{-1})

Outline

- Strong Component
- Problems

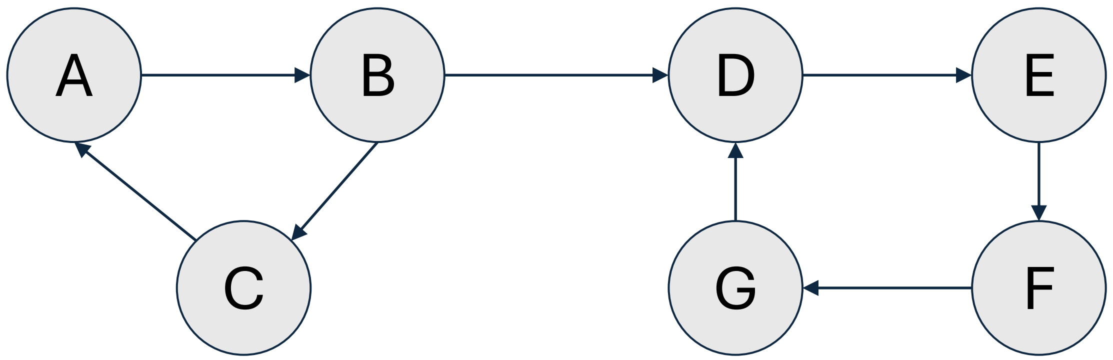
Strong Component

Claim: Fixing a DFS traversal of any directed graph G. Each strong component C of G has exactly one node that does not have a parent in G.

Proof:

- Let C be an arbitrary sc of G. Consider a path v ∈ C to w ∈ C.
- Every vertex in this path can reach w and it can also reach C.
- Similarly every vertex in this path cab reached from v and can also be reached from any vertex in C.
- So every vertex in this path is also in C.
- Let v ∈ C is the earliest vertex, i.e., start(v) is lowest among all vertices ∈ C.
- So, at the call DFS(v) all vertex in C were not visited, so any w ∈ C is a descendant of v. So except v, parent of every vertex in path v to w is in C.

Strong Component



SC(G) is equal to SC(reverse(G))

Strong Component

$s(x) < s(v) < f(x) < f(v)$

<u>Claim:</u> The last vertex in any post ordering of directed a graph lies in a source component of G.

<u>Proof:</u>

- Fix a DFS traversal in G, let v is the last vertex in the postordering. So v is the root of one of the tree in the resultant forest. So, v is the root of sc C.
- For any vertex w, if $s(v) < s(w) < f(w) < f(v)$, then w is a descendant of v.
- Let x be a vertex that is in the source component of G. Consider edge $x \to y$ such that $x \notin C$ and $y \in C$.
- x can reach y, y to v, so x can reach v. As v is the root, $s(v) < s(y) < f(y) < f(v)$.
- As there is an an edge $x \to y$, so $s(y) < f(x)$. So, $s(v) < f(x)$.
- But $f(x) < f(v)$, so $s(v) < s(x) < f(x) < f(v)$.
- Hence x is a descendant of v. So v can reach x. So, $x \in C$.

Algorithm

Input: Graph G
Output: Compute strong components.

1. Compute postordering(G) and store it in a stack.
2. Compute G^{-1}.
3. Pop vertex from stack and save nodes reachable from that vertex in G^{-1}.
4. Remove the all the reachable vertices from the stack and repeat 3.

Outline

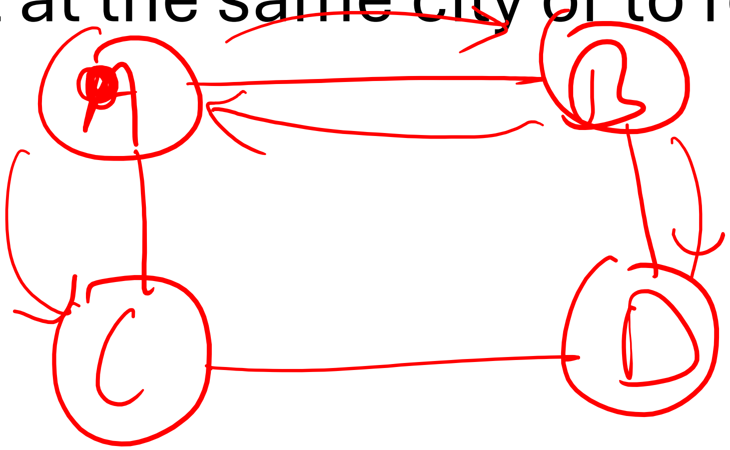
- Strong Component
- Problems
- Minimum Spanning Tree

Problems

1. Consider a directed graph G, where each edge is colored either red, white, or blue. A walk in G is called a French flag walk if its sequence of edge colors is red, white, blue, red, white, blue, and so on. Describe an algorithm to find all vertices in G that can be reached from a given vertex v through a French flag walk.

Problems

2. Suppose there are n cities connected by highways (not necessarily there is a straight road between every pair of cities). Suppose two of your friends started exploring by cars and in every step each of them can move from city A to city B if the cities have a direct road between them.
   Design an algorithm to compute minimum number of steps needed for them to meet at the same city or to report that no such paths are possible.



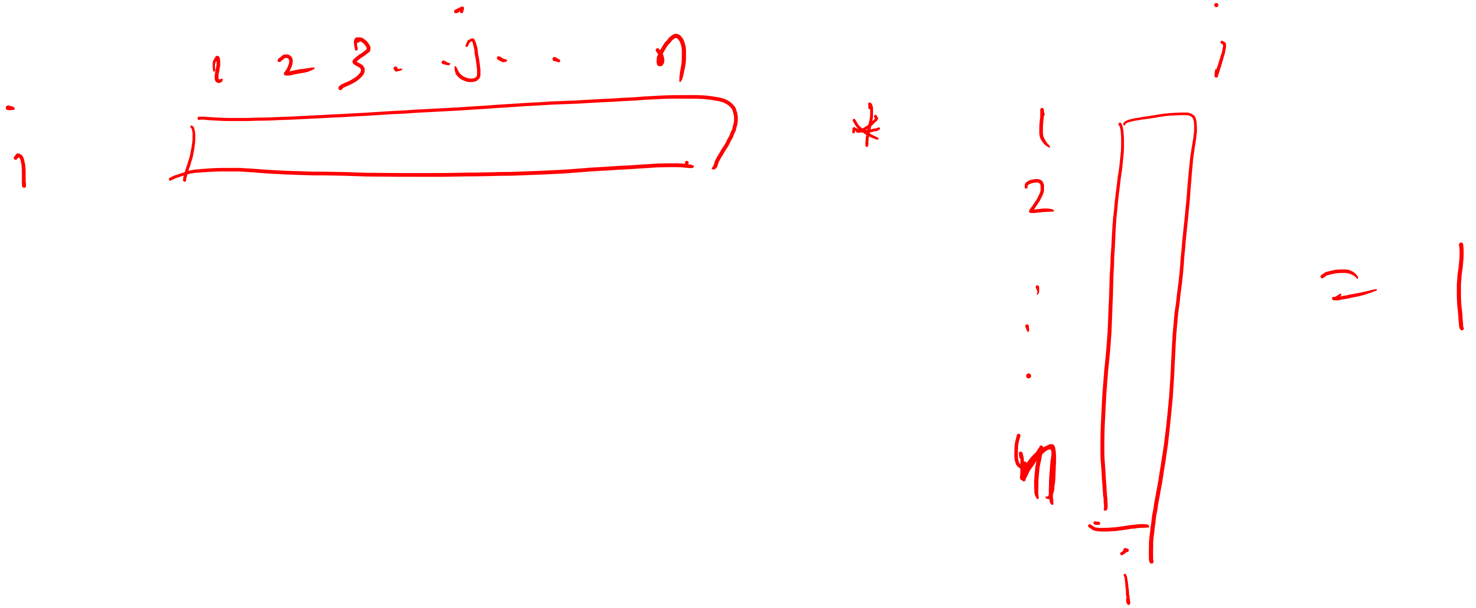
Problems

3. In a city the department of police made every street one way. Despite of complaints and confusion from people the mayor claims that it is possible to legally drive from one intersection to another. Design an algorithm to verify or refute the mayor’s claim.

Problems

4. Given a directed graph G, design an algorithm to compute number of 2k length cycle for 2 < k < n/2.

$$A \longrightarrow A^2 = (A^T A)_{ij}$$



Outline

- Strong Component
- Problems
- Minimum Spanning Tree

Minimum Spanning Tree

Let G = (V, E, w) be a weighted graph, w: E → R, i.e., w(e) is a real value.

Minimum spanning tree of G is defined by a tree T such that it minimizes the tree weight defined as follows.

$$w(T) := \sum_{e \in T} w(e)$$

Is minimum spanning tree is always unique for a given G = (V, E, w)?

Reference

Slides

Jeff Erickson Chp-6.5 & 6.6