Last lec.: Strongly connected Components (SCC)

Given a directed graph $G = (V,E)$

Find a partition of its vertex set s.t.

- $u, v \in V$ are in the same partition
- If they are strongly connected $\Rightarrow u \sim v \& v \sim u$

**Example:**

![Diagrams]

Meta-graph is acyclic (DFS).

$\exists i, j, e^3, \{a, b, c, d\}^2, \{e, h\}^3, \{f, i\}^3, \{k\}^3$

First idea:

1. Find a vertex $u$ in the source component of the meta-graph
2. Find $u$'s component
3. Remove & repeat.

Q: How to find a vertex in the source component?

1. Run DFSAll($G$)
2. $u$ = vertex of largest finish order
1. Run \( \text{DFS} \) on the directed graph.

2. Pick \( u \) = vertex of largest finish time.

**Proof sketch:**

- Not possible
- \( u, v \) are in the same connected component.

**Q:** How to find \( u \)'s component?  
- All vertices \( v \) that can reach \( u \) are in \( T_u \)
- Run DFS starting at \( u \) using "in-coming" edge

**Q:** How to remove \( u \)'s component & repeat?
- Do nothing! Just pick vertex in \( V \setminus \text{SCC}(u) \) of each

\( G_r : G \) as every edge direction is reversed for first time.

**Final algo:**

1. Run \( \text{DFSAll}(G) \), order the vertices in decreasing order of finish time.

2. Run \( \text{DFSAll}(G_r) \), while preserving vertices in the above order whenever there is a choice.

3. Output trees from step 2 as connected components.
3. Output trees from step 2 and (co-ordinates.

Running Time: \( O(m+n) \).

Exe: Run the algo on the above example.

**Shortest Paths:**

Given a directed graph \( G = (V,E) \), \( s, t \in V \)

\[ w: E \rightarrow \mathbb{R}_+ \]  (e.g. Google maps)

Find a path \( P \) from \( s \) to \( t \)

\[ \min \sum_{e \in P} w(e) \]

Special case 1: Unweighted (\( w(e) = 1 \), \( \forall e \in E \))

BFS(s). level \( (u) \) is the shortest path length from \( s \) to \( u \).

Solves Single Source Shortest Path (SSSP)

(finds \( \text{mindist}(s,u) \), \( \forall u \in V \)).

Special case 2: DAG (weighted).

Dynamic Programming. (solve SSSP)

- Define subproblem: \( \forall v \in V \)

\[ d(v) = \text{mindist}(s, v) = \text{shortest path length from } s \text{ to } v. \]

- Ans: \( d(t) \)

- Base case: \( d(s) = 0 \)

- Recursive Formula:

\[ d(v) = \min \{ d(u) + w(u,v) \} \]
Recursive Formula:
\[ d(v) = \min_{u \in \text{successors}(v)} d(u) + w(u,v) \]

Evaluation Order: Topological Sort

Running Time: \( O(m+n) \)

Note:
- Works only for DAGs.
- Also works w/ negative weights!

Dijkstra's Algo (1953):
- Assume no negative weights: \( w(e) \geq 0 \), \( e \in E \).
- Solve SSSP.
  - Compute \( d[v] = \text{shortest-path length} \), \( v \in V \)
  - from \( s \) to \( v \),

Greedy!

Idea: Consider vertices in increasing order by distance from \( s \).

Observations:
- Shortest path
Observations:

1. Shortest path

\[ S \xrightarrow{v} u \xrightarrow{0} v \]

Shortest path from \( s \) to \( u \).

2. Is the next nearest vertex to \( s \)?

\[ \text{next nearest vertex} \]

\[ \text{dist}(s, v') < \text{dist}(s, v) \]

(If \( w(e) > 0 \) for)

Next nearest vertex

has to be "adjacent" to a vertex in \( S \).

High level logic of algo:

1. Maintain the set \( S \) as "known vertices".
2. \( S = \{ s \} \), \( d[s] = 0 \)
3. \( \text{while} \ S \neq V \)

\[ O(m) \]

4. Pick edge \((u, v) \in E\) s.t. \( u \in S \), \( v \notin S \)

5. Minimize \( d[u] + w(u, v) \)

6. Set \( d[v] = d[u] + w(u, v) \)

7. Insert \( u \) to \( S \).

\[ \text{Algorithm} \]
Running Time: \( O(m \log n) \).

Ex: 

\[
\begin{align*}
&d = 0 \\
&d = 5 \\
&d = 11 \\
&d = 32
\end{align*}
\]

\[
\begin{align*}
(a, b) & : 0 + 5 \\
(a, c) & : 0 + 9 \\
(b, e) & : 5 + 28 \\
(b, d) & : 5 + 7 \\
(c, d) & : 9 + 2 \\
(b, e) & : 5 + 28 \\
(d, e) & : 11 + 21
\end{align*}
\]
Proof of correctness:

Claim: Assume $d[u]$ value are correct for $u \in S$.

If $(u,v)$ is an edge from $u \in S$ to $v \in V \setminus S$ with smallest $d[v] + w(u,v)$ value.

Then $\text{mindist}(s,v) = d[v] + w(u,v)$.

($\leq$) There is a path from $s$ to $v$.

$$S \rightarrow d[v] \rightarrow w(u,v) \rightarrow v$$

$$\text{mindist}(s,v) \leq d[v] + w(u,v)$$

($\geq$) Take shortest path $p^*$ from $s$ to $v$.

$$\text{mindist}(s,v) = \text{length}(p^*) = d[u'] + w(u',v) + \ell$$

$$\geq d[v] + w(u,v)$$

$$> d[u] + w(u,v)$$
**Efficient Implementation (using priority Queue):**

1. \( \textsf{Q} = V \setminus \emptyset \)
2. \( \text{key}[v] = \infty \), \( \forall v \in V \setminus \emptyset \) \( \text{key}[\emptyset] = 0 \)
3. while \( \text{Q} \neq \emptyset \) 
   1. Pick \( u \in \text{Q} \) with smallest key.
   2. \( d[v] = \text{key}[v] \).
   3. Remove \( u \) from \( \text{Q} \).
   4. for each out-neighbor \( y \) of \( v \)
      1. if \( y \in \text{Q} \) & \( d[v] + w(v, y) < \text{key}[y] \)
         1. then \( \text{key}[y] = d[v] + w(v, y) \); \( \text{pred}[y] = v \).

**Diagram:**

**Before:**

**After:**
<table>
<thead>
<tr>
<th>Running Time: No data structure for $O$.</th>
<th>Priority Queue for $O$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>line 4: $O(n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>line 6: $O(1)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>line 8: $O(1)$</td>
<td>$O(\log n)$</td>
</tr>
</tbody>
</table>

Total: $O(n^2 + \sum_{v \in V} \text{out-deg}(v) \cdot O(1))$

$= O(n^2 + m) = O(n^2)$

$= O(m \cdot \log n + m \cdot \log m)$

$= O((n + m) \log n)$