1 Graphs and Graph Algorithm

A graph $G = (V, E)$ is a set of edges $E$ and $E \subseteq V \times V$

Definition 1. path: $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow \cdots \rightarrow v_n$

simple path: visit each vertex $\leq 1$ time

degree: number of edges in one specific vertex

out degree and in degree: for directed graph, number of edges start from the vertex and number of edges end in the vertex

cycle: path that ends at start $v_1 = v_T$ has length $> 0$

Proposition 2. $\sum_{v \in V} \text{deg}(v) = 2|E|$

Definition 3. connected: vertices are connected if $\exists$ path from one to the others

graph connected $\Leftrightarrow$ all pairs of the other vertices connected

tree: connected acyclic graph (without cycles) $n$ vertices $n - 1$ edges connected

DAG: directed acyclic graph

A directed graph is strongly connected if $\forall v_1, v_2 \exists$ path $v_1 \rightarrow v_2$ and $v_2 \rightarrow v_1$

$(v_1, v_2) : v_1 \rightarrow v_2, v_2 \rightarrow v_1$

distance $(u, v) =$ length of shortest path $u \rightarrow *v$

$$
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5
\end{array}
\end{array}
\end{array}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5
\end{array}
\end{array}
\end{array}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5
\end{array}
\end{array}
\end{array}
\end{array}
$$

Figure 1.1: Example of undirected graph
2 Representing Graphs

There are two standard ways to represent a graph:

- collection of adjacency lists: preferred by sparse graph, i.e. $|E| \ll |V|^2$
- adjacency matrix: preferred by dense graph, i.e. $|E| \approx |V|^2$

Assume $n = |V|$, $m = |E|$, then we have following propositions:

Proposition 4. for each vertex $v \in V$ (hashtable) $O(n)$

Proposition 5. for each $(v_1, v_2) \in E$ (hashtable) $O(m)$

Proposition 6. if $\exists (v_1, v_2) \in E$ (hashtable) $O(1)$ (average)

Proposition 7. for all $u$ s.t. $(u, v) \in E$ $O(\deg v)$

3 Depth-First Search/Breadth-First Search

We denote the depth-first search from a vertex $v$ as $dfs(v)$, and denote $v.first$ as the time of first visit, $v.last$ as the time of last visit. Then the algorithm can be summarized as follows:

```
for v ∈ V  
v.first = ⊥  
current_index = 0  
dfs(v)  
{
  v.ccid = connected_component_id  
v.first = current_index ++  
  for each edge(v, v')  
  {
    if v'.first = ⊥
      dfs(v')
    v.last = current_index ++;
  }
```
First three lines: initialize $O(n)$
later lines: $\sum O(\deg v) = O(m)$

**Proposition 8.** After $dfs(v)$ returns every vertex $v'$ in the connected component of $V$ will have $v'.\text{first} \neq 1$.

Since the graph might have several connected components, we have to search all the vertices to ensure all the several connected components are searched. The process is summarized as follows:

\[
dfs(G)
\]
\[
\text{connected\_component\_id} = 0
\]
\[
\text{for each } v \in G
\]
\[
\text{if } v.\text{first} = \bot
\]
\[
\{\text{dfs}(v)\text{connected\_component\_id}++\}
\]

Claim: labels each vertex of ccid.
This algorithm is $O(n)$.
Total time is $O(m + n)$.

![Figure 3.1: Example of Depth First Search](image)