**Depth First Search**

\[ G = (V, E) \]

Graph

\[ V = \{ v_1, v_2, v_3, \ldots, v_n \} \]

Set of vertices

\[ |V| = n \]

Order of the graph

\[ E = \{ (u, v) \mid u, v \in V \} \]

Set of edges

Ordered pair

Directed

Unordered

Undirected

\[ (u, v) = (v, u) \]

\[ |E| = m \]

Size of the graph

Simple graphs \( \Rightarrow \) no self loops, no multi graph
\[ G = (V, E) \]

"binary representation"

weighted graph,

\[ G = (V, E, W) \]

\( V = \text{vertices} \)

\( E = \text{edges} \)

\( W(e) \in \mathbb{R} \) is the weight on edge \( e = (u, v) \)

Probabilistic graph
Labels vs. Unlabeled

There are "labels" on the vertex and edge

\[ G = (V, E, L_v, L_e) \]

\[ L_v (v) = \text{label for } v \]

\[ L_e (e) = \text{label on an edge } e \]

Graph matching / isomorphism

Representation

Adjacency matrix \( A \)

\[ G = (V, E) \]

\[ |V| = n \]

\[ |E| = m \]

\[ 1 \quad 0 \quad 1 \]

\[ 0 \quad 1 \quad 2 \]

\[ \begin{pmatrix}
 0 & 1 & \cdots & 0 \\
 1 & 0 & \cdots & 0 \\
 \vdots & \vdots & \ddots & \vdots \\
 0 & 0 & \cdots & 0
\end{pmatrix} \]
Neighbors of a vertex $v$:

$$ N(v) = \{ u \mid (v, u) \in E \} $$

Set of all "adjacent" to $v$.

Existence of edge $e_{uv}$:

$$ N(3) = \{ 1, 3, 6 \} $$

Undirected graph:

Directed graph:

Weighted adjacency matrix:

$n \times n$, symmetric binary matrix.
existence of an edge
\[
\begin{pmatrix}
0 & 1 \\
1 & 0
\end{pmatrix}
\]
for adjacency matrix \( A \) take \( O(n) \) time

\[
A = \begin{pmatrix}
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0
\end{pmatrix}
\]

\[
N(i, j) = 1, \ldots, 3
\]

\[
A = n \times n \text{ matrix} = O(n^2)
\]

has to record
both the existence
& absence of an edge

1) Dense graph

\[
G = (V, E)
\]

\[
density = \frac{|E|}{\text{ # of possible edges}} = \frac{m}{\binom{n}{2}} \leq 1
\]

Undirected: \( |V| = n \)

\[
\binom{n}{2} = \frac{n(n-1)}{2} \approx O(n^2)
\]
A graph is dense if \( n = O(n^2) \)

2) Real-world: Graphs are sparse

\[ m = O(n) \]

A node/vertex is connected to some of the other nodes.

\[ \text{degree}(v) = \text{number of nodes that are adjacent to } v \]

\[ = |N(v)| \leq \text{size of neighbors} \]

E.g., Facebook \( n = \text{billion} \)

\[ \text{avg degree}(v) \leq 10^6 \]

\[ \text{avg degree}(v) \leq 10 \text{ or } 100 \]

\( O(n^2) \) edge complexity
For sparse graphs

Adjacency list representation

\[ E = m \]

\[ \mathbb{N}(v) \]

| \text{space} | \mathcal{O}(m) |

\[ \mathcal{O}(n) \text{ for sparse graph,} \]

Walk versus Path
Walk is any sequence of alternating vertices & edges in a graph

Walk:

\[ 2 \rightarrow e_2 \rightarrow 3 \rightarrow e_3 \rightarrow 1 \rightarrow e_3 \rightarrow 2 \rightarrow e_4 \rightarrow 1 \]

\[ 2 \rightarrow 3 \rightarrow 1 \rightarrow 2 \rightarrow 4 \]

(2, 3, 1, 2, 4)

Path: no duplicate vertices allowed, except for the start & end

Cycle: is a path with start & end

\[ 1 \rightarrow 2 \rightarrow 3 \rightarrow 1 \]

(1, 2, 3, 1)

\[ 2 \rightarrow 1 \rightarrow 2 \rightarrow 0 \]

\[ \text{path} \]

DF-S: Depth First search

\[(1)\]
DFS explore (u)
visited (u) = True
for x ∈ N(u):
  if not visited (x):
    DFS explore (x)

N(3) = \{2, 5\}

DFS → find all reachable vertices from u
∃ a path from u to v

Disconnected graph

Connected components:
\[ CC_1 = \{ 1, 2, 3, 4 \} \]

\[ CC_2 = \{ 5, 6, 7 \} \]

**DFS to find Connected Components**

\[
CC(G) = \begin{cases} 
\text{for all } u \in V \\
\quad \text{visited}(u) = 0 \\
\quad \text{id} = 0 \\
\quad \text{for all } u \in V \\
\quad \quad \text{if visited}(u) = 0 : \\
\quad \quad \quad \text{DFS explore}(u, \text{id}) \\
\quad \quad \text{id} = \text{id} + 1 
\end{cases} \\
O(n)
\]

Recursive calls \( \sim O(n) \)

\[ \Rightarrow \text{cost of each call?} \]

\[ \text{how many neighbors are there across all nodes?} \]

\[ \text{DFS explore}(u, \text{id}) \]

\[
\begin{align*}
\text{visited}(u) &= 1 \\
\text{id}(u) &= \text{id} \\
\text{for all } x \in \text{N}(u) : \\
\quad \text{if visited}(x) &= 0 \quad \text{DFS explore}(x)
\end{align*}
\]
Every edge will be checked exactly 2 times.

DFS \( O(n+m) \)

\( O(|V|+|E|) \)

Linear time algorithm in the graph "size".

Aggregate analysis: across all nodes

1) Each node is visited once
2) We look at all its neighbors

DFS with pre & post number

Order of visit in DFS exploration
A directed graph.

Nodes labeled with ranges:
- Node 2: [0, 9]
- Node 3: [1, 8]
- Node 1: [2, 3]
- Node 4: [4, 7]
- Node 5: [5, 6]

DFS pre-order:
- 2
- 0
- 9
- 1
- 2
- 3
- 4
- 5
- 6

DFS post-order:
- u: [x, y]

Reachability can be checked in constant time after DFS.

Q: 'Is there a path from 2 to 1?'

\([0, 9] \leq \[2, 3]\)

Since \([2, 3] \leq [0, 9]\)

true