Graph Algorithms

Graph $G = (V, E)$

- $V$: set of vertices
- $E$: set of edges

$|V| = n$
$|E| = m$
$(n-1) \leq m \leq n^2$

$V = \{a, b, c, d, e\}$
$E = \{(a, b), (c, a), (b, d), (a, d), (d, e), (e, b), (c, d)\}$

Application: Facebook graph, social network, internet

* Basic concepts: path, connected, cycles.

* Representation:
  - Adjacency Matrix
    
    $A = \begin{bmatrix}
    0 & 1 & 1 & 0 & 0 \\
    1 & 0 & 1 & 1 & 0 \\
    1 & 1 & 0 & 0 & 0 \\
    0 & 1 & 0 & 0 & 1 \\
    0 & 0 & 0 & 1 & 0 \\
    \end{bmatrix}$

    (directed and undirected, matrix representation)
(id 6 underscored)
A is symmetric matrix

\[ \text{space} = O(n^2) \quad \text{look up time} \quad O(1). \]

- Adjacency list

\[
\begin{array}{c|ccc}
\text{a} & \text{b} & \text{d} \\
\text{b} & \text{d} & \\
\text{c} & \text{d} & \text{a} \\
\text{d} & & \\
\text{e} & & \\
\end{array}
\]

\[ \text{Adj}(u) = \{v \mid (u, v) \in \mathcal{E} \} \]

\[ \text{space} : O\left( \sum_{u \in V} |\text{Adj}(u)| \right) \]

\[ = O(m + n) \]

(it space graph)

\[ m \ll n^2 \]

\* Basic que:
- if \( G \) a path from \( s \) to \( t \), on \( bel^m s \ t \)
- if \( G \) is connected?
- vertices reachable from \( s \)?

\* Basic Search Algo:
- Breadth First Search (BFS)
- Depth First Search (DFS)
Eg. Tree

BFS

DF

level 0
1
2
3

1

2

3

4

5

6

7

8

9

10

1

2

3

4

Discover order:
- Breadth traversal

Finish order:
- Post order.

*Graph (Extension)*

start

BFS

NO forward edges
in BFS tree.

DFS

descendants.

*Non-tree edges:*

- Back edges: edge from a node to one of its ancestors.
- Forward edges: edge to a descendant.
- Cross edges: all other non-tree edges.
Implementation: \( \text{BFS}(G, s) \)

// idea 1: Mark visited vertices.

// idea 2: Use a data structure \( Q \).

\( Q \leftarrow \) 1) for \( u \in V \), do unmark \( u \).

2) Insert \( s \) in \( Q \). Mark \( s \). level \([s]\) = 0

3) while \( Q \neq \emptyset \) do

4) remove a vertex \( u \) from \( Q \).

5) for each \( v \in \text{Adj}(u) \) do if \( v \) is unmarked

6) insert \( v \) in \( Q \). Mark \( v \). parent \([v]\) = \( u \). level \([v]\) = level \([u]\) + 1

Runtime: steps \( \leq O(1 + \text{Adj}(u)) \)

Total time \( O(\sum_{u \in V} |\text{Adj}(u)|) + O(n) \)

= \( O(m + n) \)

Global time = 1

DFS(\( G, u \) )

// similar, with different data structure: stack or recursion.

\[ \text{DFS}(G, u) \]
1) Mark u, discovered \([u] = \text{time} + t\).
2) for \(v \in \text{Adj}(u)\) do \\
3) if \(v\) is unmarked \\
4) \quad \text{DFS}(G, v) \\
5) \quad \text{Parm} [v] = u \\
6) \quad \text{Finished} \[u] = \text{time} + t

\[a1\]: Shortest path distance from s to t.