CSE 421: Introduction to Algorithms

Induction - Graphs

Shayan Oveis Gharan
Undirected Graphs $G=(V,E)$

Disconnected graph
Multi edges
Isolated vertices
Self loop
Simple
Graphs don’t Live in Flat Land

Geometrical drawing is mentally convenient, but mathematically irrelevant:

4 drawings of a single graph:
Directed Graphs

[Diagram of a directed graph with labeled nodes 1 to 13, showing directed edges and highlighting a self-loop at node 12 and a multi-edge between nodes 11 and 13.]
Terminology

- **Degree of a vertex**: \# edges that touch that vertex
  \[ \text{deg}(6) = 3 \]
  \[ \text{deg}(10) = 2 \]

- **Connected**: Graph is connected if there is a path between every two vertices

- **Connected component**: Maximal set of connected vertices

[Diagram of a graph with labeled vertices and edges, indicating connected components and degrees.]
Terminology (cont’d)

• **Path**: A sequence of distinct vertices s.t. each vertex is connected to the next vertex with an edge

• **Cycle**: Path of length $> 2$ that has the same start and end

• **Tree**: A connected graph with no cycles

3, 6, 5, 3 not a path
Degree Sum

Claim: In any undirected graph, the number of edges is equal to \((1/2) \sum_{\text{vertex } v} \deg(v)\)

Pf: \(\sum_{\text{vertex } v} \deg(v)\) counts every edge of the graph exactly twice; once from each end of the edge.

\(|E|=8\)

\[\sum_{\text{vertex } v} \deg(v) = 2 + 2 + 1 + 1 + 3 + 2 + 3 + 2 = 16\]
Odd Degree Vertices

Claim: In any undirected graph, the number of odd degree vertices is even

Pf: In previous claim we showed sum of all vertex degrees is even. So there must be even number of odd degree vertices, because sum of odd number of odd numbers is odd.

4 odd degree vertices
3, 4, 5, 6

Sum of odd number of odd numbers is odd.
Degree 1 vertices

Claim: If G has no cycle, then it has a vertex of degree $\leq 1$
(So, every tree has a leaf)

Pf: (By contradiction)
Suppose every vertex has degree $\geq 2$.
Start from a vertex $v_1$ and follow a path, $v_1, ..., v_i$ when we are at $v_i$ we choose the next vertex to be different from $v_{i-1}$. We can do so because $\text{deg}(v_i) \geq 2$.
The first time that we see a repeated vertex ($v_j = v_i$) we get a cycle.
We always get a repeated vertex because $G$ has finitely many vertices.

Diagram:

```
  v1---v2---v3---v4---v5
     |       |       |
     |       |       |
     |       |       |
     |       |       |
     |       |       |
     |       |       |
     |       |       |
     |       |       |
     |       |       |
     |       |       |
```
**Claim**: Show that every tree with $n$ vertices has $n-1$ edges.

**Pf**: By induction.

**Base Case**: $n=1$, the tree has no edge

**IH**: Suppose every tree with $n-1$ vertices has $n-2$ edges

**IS**: Let $T$ be a tree with $n$ vertices.

So, $T$ has a vertex $v$ of degree 1.

Remove $v$ and the neighboring edge, and let $T'$ be the new graph.

We claim $T'$ is a tree: It has no cycle, and it must be connected.

So, $T'$ has $n-2$ edges and $T$ has $n-1$ edges.
Let $G = (V, E)$ be a graph with $n = |V|$ vertices and $m = |E|$ edges.

**Claim:** $0 \leq m \leq \binom{n}{2} = \frac{n(n-1)}{2} = O(n^2)$

**Pf:** Since every edge connects two distinct vertices (i.e., G has no loops) and no two edges connect the same pair of vertices (i.e., G has no multi-edges) it has at most $\binom{n}{2}$ edges.
Sparse Graphs

A graph is called sparse if $m \ll n^2$ and it is called dense otherwise.

Sparse graphs are very common in practice
• Friendships in social network
• Planar graphs
• Web braph

Q: Which is a better running time $O(n + m)$ vs $O(n^2)$?
A: $O(n + m) = O(n^2)$, but $O(n + m)$ is usually much better.
Storing Graphs (Internally in ALG)

Vertex set $V = \{v_1, \ldots, v_n\}$.

Adjacency Matrix: $A$

- For all, $i, j, A[i, j] = 1$ iff $(v_i, v_j) \in E$
- Storage: $n^2$ bits

Advantage:
- $O(1)$ test for presence or absence of edges

Disadvantage:
- Inefficient for sparse graphs both in storage and edge-access
Storing Graphs (Internally in ALG)

**Adjacency List:**
O(n+m) words

**Advantage**
- Compact for sparse
- Easily see all edges

**Disadvantage**
- No O(1) edge test
- More complex data structure
Storing Graphs (Internally in ALG)

Adjacency List:
$O(n+m)$ words

Advantage
- Compact for sparse
- Easily see all edges

Disadvantage
- No $O(1)$ edge test
- More complex data structure
Graph Traversal

Walk (via edges) from a fixed starting vertex $s$ to all vertices reachable from $s$.

- Breadth First Search (BFS): Order nodes in successive layers based on distance from $s$
- Depth First Search (DFS): More natural approach for exploring a maze; many efficient algs build on it.

Applications:
- Finding Connected components of a graph
- Testing Bipartiteness
- Finding Aritculation points
Breadth First Search (BFS)

Completely explore the vertices in order of their distance from $s$.

Three states of vertices:
- Undiscovered
- Discovered
- Fully-explored

Naturally implemented using a queue
The queue will always have the list of Discovered vertices
BFS implementation

Global initialization: mark all vertices "undiscovered"

BFS(s)

mark s "discovered"
queue = { s }
while queue not empty
    u = remove_first(queue)
    for each edge {u,x}
        if (x is undiscovered)
            mark x discovered
            append x on queue
mark u fully-explored