Next few lectures: Graphs! "vertices" graph is a pair \( G = (V, E) \)

Directed: \( E \subseteq V \times V \) (if \( (a, a) \) not allowed, then we call graph "simple")

Undirected: \( E \) is a set of unordered pairs from \( V \)

Examples:

1. Road network. Vertices are intersections. Edges are road segments connecting intersections. (would be directed, also maybe "weighted")
2. Social network: Twitter/IG: directed FB: undirected
Represent graph to computer

For vs: \( V = \{ 1, \ldots, n \} \)

Represent edges:
1. Adjacency Matrix rep
2. Adjacency List rep.

\( A = \begin{bmatrix} 1 & \cdots & \cdots & \cdots & 1 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & \cdots & \cdots & \cdots & 1 \end{bmatrix} \)

\( A_{ij} = \begin{cases} 1, & \text{if } (i,j) \in E \\ 0, & \text{if } (i,j) \notin E \end{cases} \)

(or in weighted graphs)

\( A_{ij} = \begin{cases} \sum_{i,j} W_{ij}, & \text{if } e \in E \\ \infty, & \text{if } e \notin E \end{cases} \)
(2) Adjacency List

Represent $E$ as an array $B$ of linked lists.

$B[ij]$ is a linked list containing all $j$ such that $(ijj) \in E$

Examples:

\[ B = \]

\[ \text{Adj. matrix} \quad \text{Adj. List} \]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>$n^2$ bits</td>
<td>$\Theta(m+n)$ words</td>
</tr>
<tr>
<td>$(u,v) \in E$</td>
<td>$O(1)$ time</td>
<td>$\Theta(du+1)$ time</td>
</tr>
</tbody>
</table>

$du$ = "degree" of $u$ = number of $v$ such that $(u,v) \in E$

n := |V|

m := |E|
Running Algorithms on graphs

**Graph Exploration**  Depth First Search (DFS)

```python
def DFS(V, E):
    # Global variables
    global clock = 1
    global visited = boolean [E] (initially all False)
    global preorder, postorder = list [E]

    for v in V:
        if !visited[v]:
            explore(v)
            clock += 1

    for (v, w) in E:
        if !visited[w]:
            explore(w)
            postorder[v] = clock + 1
```

# Times if !visited[v] triggered = # of CC's.
Claim: A DFS search from vertex $u$ explores exactly the set of vertices $v$ s.t. $\exists$ path from $u$ to $v$.

Let $\text{explore}(v) \Rightarrow \exists \text{ path from } u \text{ to } v$ in recursion tree from $u$ to $v$ is a path in the graph.

For the sake of contradiction, if path will explore $v$ which doesn't get explored.

Let $x_j$ be first vertex on path which isn't explored. $\Rightarrow x_{j+1}$ would be explored recursively. Contradiction.
Runtime of DFS I

Total time: $\Theta(n) + \sum_{uv \in V} (\text{the to enumerate neighbor of } u)$

Adj matrix: $\Theta(n^2)$ time

Adj list: $\Theta(n) + \Theta(\sum_{uv \in V} 1)$

Each edge counted exactly once

$= \Theta(m+n)$
DFS has many applications:
- reachability
- identifying connected components
- articulation points (set of vertices whose removal would disconnect graph)
- finding biconnected / triconnected components.
- **Strongly connected components. (Will be next week)**
- planarity testing
- isomorphism of planar graphs
Claim: The set of intervals $[\text{pre}(u), \text{post}(u)]$ are nested or disjoint.

- $[[ ]] \checkmark$
- $[ [ ] ] \checkmark$
- $[ [ ] ] \times \text{(impossible)}$
- $[ [ ] ] \checkmark$
- $\ldots$
Typed of edges in graph

1. Tree edge: traversed during DFS
2. Forward edge: goes from an ancestor to a descendant in DFS tree, and is not a tree edge
3. Back edge: goes from descendant to ancestor in DFS tree
4. Cross edge: all other edges of G.

Claim: \( \text{post}(u) < \text{post}(v) \) iff \((u,v)\) is a back edge.

Claim: G has a cycle iff it has a back edge.