CS 180 - Graphs

Announcements

- I'll post room for review session next Friday.

- Program due Sunday by 11:59 pm.
Recap:

Topics
- Basic C++ and run-times
- Stacks
- Queues
- Lists
- Vectors
- Sorting
- Binary Trees
- BSTs
- AVL trees
- Huffman Trees
- Hashing
- Graphs
- Heaps

represented on worksheet
Graphs

A graph $G = (V, E)$ is a set containing $V$ and $E$

$V = \text{vertices}$
$E = \text{edges (or pairs of vertices)}$

$|V| = n$
$|E| = m$

$e_i = \{v_i, v_j, v_k\}$
Examples:

- routes - road networks
- specialise
- Facebook
- games
  - (3-dim meshes)
- internet
- collaboration
Definitions:

- $G$ is **undirected** if every edge is an unordered pair, so $\{u, v\} = \{v, u\}$.

- $G$ is **directed** if every edge is an ordered pair $e = (u, v)$.

$V_1$ is connected to $V_2$. The tail of $u$ and head of $v$.
Definitions:

- The degree of a vertex \( v \), \( d(v) \), is the number of adjacent edges.

- A path \( P = v_1, v_2, \ldots, v_k \) is a set of vertices such that \( \{v_i, v_{i+1}\} \in E \) (usually no repeated vertices).\[d(v)=3\]

- A path is simple if all vertices are distinct.

- A path is a cycle if it is simple except for \( v_1 = v_k \).\[\text{red is a cycle}\]
Lemma (degree-sum formula):
\[ \sum_{v \in V} d(v) = 2|E| \]

Proof:
Counting every edge twice (incidence)
How to analyze:

let $G$ have $n$ vertices.

How big can $m$ be?

$m = O(n^2)$

$\frac{m}{n} \leq \frac{n(n-1)}{2} = \binom{n}{2}$

Each vertex connects to $\leq n-1$ other vertices

$\deg\text{-sum} \Rightarrow d(v) \leq n-1$

$2m = \sum_{v \in V} d(v) \leq \sum_{v \in V} (n-1) = n(n-1)$

$2m \leq n(n-1)$
So—how to store these?

- list for each vertex
  - of edges or vertices it connects
- list of vertices
- list of edge
  - each could point to other
Vertex List:

\[ V_1 \rightarrow V_2 \rightarrow V_5 \rightarrow V_4 \]
\[ V_1 \rightarrow V_3 - V_5 \]

\[ = O(n + m) \leftarrow \leq O(n^2) \]
\[ O(n) \]

if we can "sort",
\[ O(\log n) \]
Adjacency matrices

Suppose $G$ is weighted.

Space: $O(n^2)$

Nice things: symmetric (if undirected $G$)

Worse space
Incidence Matrices

\[
\begin{array}{cccc}
V_1 & e_1 & e_2 & e_3 \\
V_2 & 1 & 0 & 0 \\
V_3 & 1 & 1 & 0 \\
V_4 & 1 & 1 & 1 \\
\end{array}
\]

Space: $O(nm) \leq O(n^3)$
Which are best?

In terms of space —

Vertex lists

Sometimes use adjacency matrices.
Dfs:

- G is connected from u to v if \( \forall u, v \in V \exists \text{ a path} \)

- The distance from u to v, \( d(u, v) \), is equal to the length of a minimum \( u, v \) path
Algorithms on graphs

Basic question: given 2 nodes, are they connected?

How to solve?

Breadth-First Search

$O(n + m)$

$u \rightarrow v$

$O(n)$
Suggestion:

Pretend we are in a maze searching for a treasure.

How do you proceed?

Depth First Search
**Recursive DFS (u):**

If u is unmarked
  mark u
  for each edge \( \{u,v\} \in E(G) \)
    Recursive DFS(v)
  endfor
end if

For s-t connectivity, call DFS(s), if it is ever marked then they are connected.
Example

DFS tree:
**Def:** A tree is a connected, acyclic graph.

A leaf in a tree is a vertex \( v \) with \( d(v) = 1 \).
Running time of DFS?